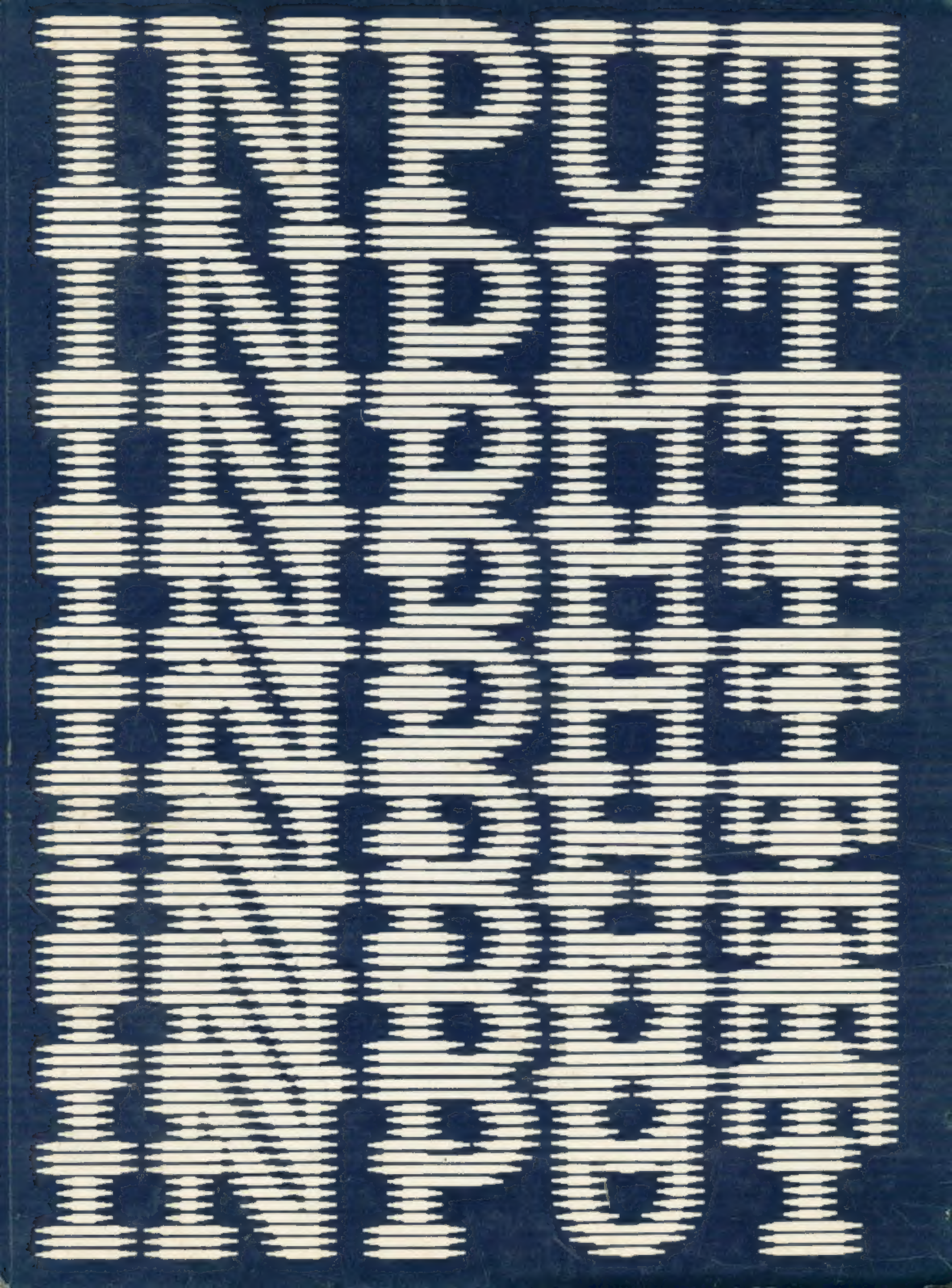

INPUT

INPUT

5


EDITOR
RICHARD CIVITA

NOVA CULTURAL

Presidente
Flávio Barros Pinto

Diretoria
Carmo Chagas, Iara Rodrigues,
Pierluigi Bracco, Plácido Nicoletto,
Roberto Silveira, Shozi Ikeda,
Sônia Carvalho

REDAÇÃO
Diretor Editorial: Carmo Chagas
Editores Executivos:
Antonio José Filho, Berta Sztark Amar
Editor Chefe: Paulo de Almeida
Editoras Assistentes: Ana Lúcia B. de Lucena,
Marisa Soares de Andrade
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio,
Grace Alonso Arruda, Monica Lenardon Corradi

Colaboradores
Consultor Editorial Responsável:
Dr. Renato M. E. Sabbatini
(Diretor do Núcleo de Informática da
Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria
em Informática Ltda., Campinas, SP
Tradução, adaptação, programação e redação:
Abílio Pedro Neto, Aluísio J. Dornellas de Barros,
Marcelo R. Pires Therezo, Marcos Huascar Velasco,
Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira
Coordenação Geral: Rejane Felizatti Sabbatini

COMERCIAL
Diretor: Roberto Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Mozol
Gerente de Propaganda e Publicidade: José Carlos Madio
Gerente de Pesquisa e Análise de Mercado:
Wagner M. P. Nabuco de Araújo

(CLC)

A Editora Nova Cultural Ltda. é uma empresa do
Grupo CLC — Comunicações, Lazer, Cultura S.A.
Presidente: Richard Civita
Diretoria: Flávio Barros Pinto, João Gomez,
Menahen M. Politi, Renê C. X. Santos,
Stélio Alves Campos

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, n.º 2000 - 3.º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda.
e impressa na Divisão Gráfica da Editora Abril S.A.

SUMÁRIO

APLICAÇÕES

CÓDIGO DE MÁQUINA

LINGUAGENS

PERIFÉRICOS

PROGRAMAÇÃO BASIC

PROGRAMAÇÃO DE JOGOS

SOFTWARE

COMPRESSÃO DE MELODIAS

- PROGRAMAÇÃO DE UMA MELODIA
- COMO COMPRIMIR OS DADOS
- DIVISÃO DA MÚSICA
- ECONOMIA DE NOTAS
- MODIFICAÇÃO DO ANDAMENTO

Amplie a capacidade musical de seu micro reduzindo os dados necessários à execução de uma melodia. As técnicas apresentadas aplicam-se à compressão de diferentes tipos de dados.

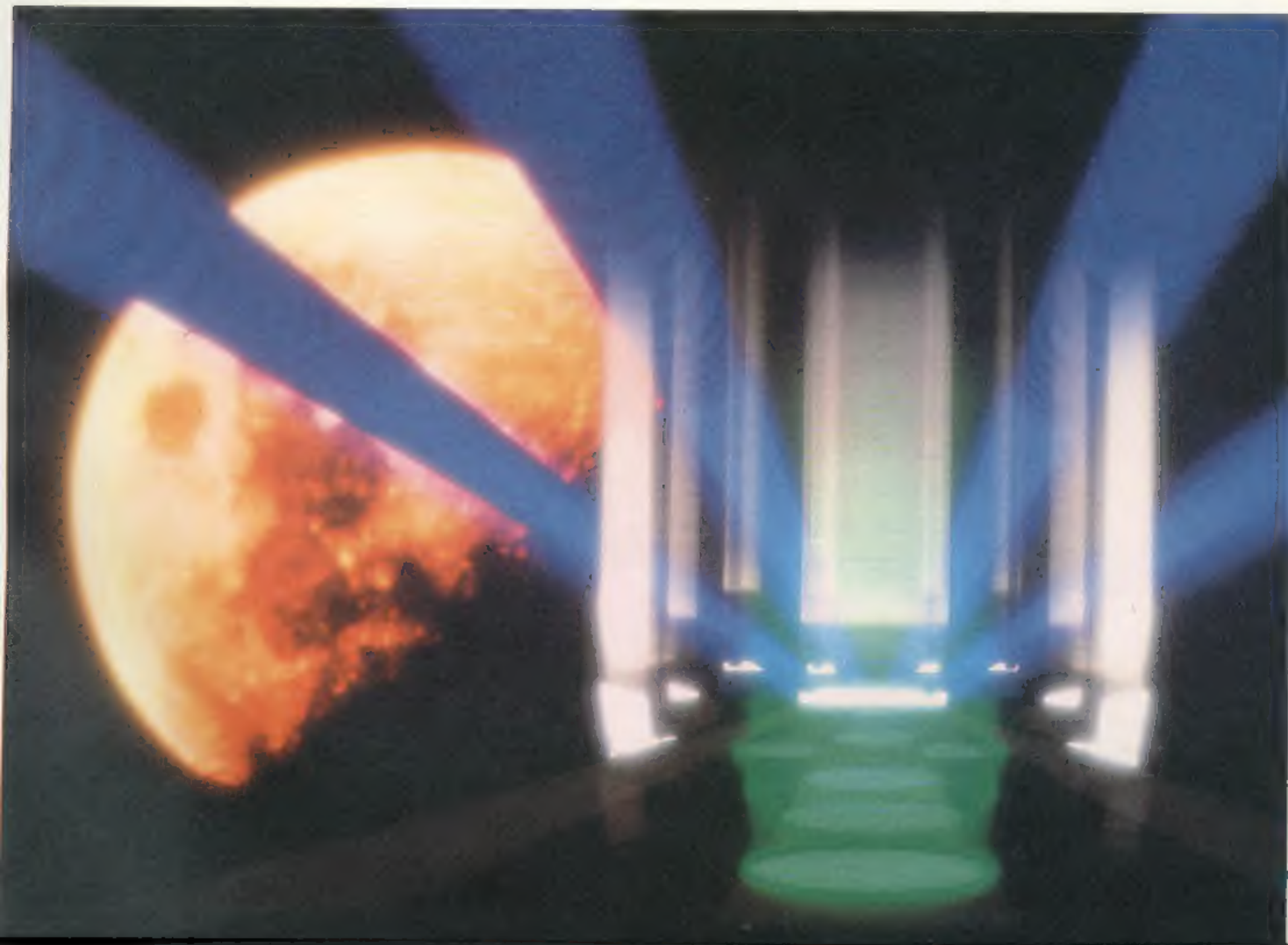
Tocar música utilizando o computador é uma experiência fascinante, principalmente quando se é o autor da peça. Existem, porém, várias dificuldades a superar, e uma delas é a quantidade de dados que um programa requer para executar determinada melodia — mais de uma tela cheia de números, na maioria dos casos. Além de serem difíceis de digitar, esses dados ocupam muito espaço na memória. Neste artigo, você verá como comprimir os dados de suas melodias dentro dos programas BASIC, de maneira que eles tomem o menor espaço possível, liberando parte da memória RAM para outro uso.

A aplicação das técnicas de compressão de dados não se restringe, naturalmente, aos programas destinados à execução de música. Estende-se a dados de todo tipo, desde que sejam repetitivos e limitados a certos valores.

UMA NOVA MELODIA

Seja qual for seu estilo, as melodias que o computador toca seguem um certo padrão que permite sua compressão. Tomemos como exemplo uma música simples, com um total de doze compassos. Para executá-la, precisaremos escrever um programa em BASIC contendo um determinado número de linhas **DATA**, como o apresentado a seguir:

```
 10 LET T=.2
 20 RESTORE 100
 30 READ D
 50 IF D=255 THEN  GOTO 20
 60 SOUND T,D
 70 GOTO 30
100 DATA 12,12,15,16,19,19,21,
19
110 DATA 12,24,22,21,19,17,16,
14
120 DATA 12,12,15,16,19,19,21,
19
130 DATA 12,24,22,21,19,17,16,
14
140 DATA 17,17,20,21,24,24,26,
```



```
24
150 DATA 17,24,22,21,19,17,16,
14
160 DATA 12,12,15,16,19,19,21,
19
170 DATA 12,24,22,21,19,17,16,
14
180 DATA 19,19,23,24,26,26,24,
23
190 DATA 17,17,20,21,24,24,20,
21
200 DATA 12,12,15,16,19,19,21,
19
210 DATA 12,24,22,21,19,17,16,
14
220 DATA 255
```



```
10 SOUND 7,56:SOUND 8,15
20 SOUND 1,0:RESTORE
30 READ D
50 IF D=255 THEN 20
60 SOUND 0,D
70 FOR T=0 TO 100:NEXT:GOTO 30
100 DATA 213,213,179,169,142,14
2,126,142
110 DATA 213,106,119,126,142,15
9,169,189
120 DATA 213,213,179,169,142,14
2,126,142
130 DATA 213,106,119,126,142,15
9,169,189
140 DATA 159,159,134,126,106,10
6,94,106
150 DATA 159,106,119,126,142,15
9,169,189
160 DATA 213,213,179,169,142,14
2,126,142
170 DATA 213,106,119,126,142,15
9,169,189
180 DATA 142,142,112,106,94,94,
106,112
190 DATA 159,159,134,126,106,10
6,134,126
200 DATA 213,213,179,169,142,14
2,126,142
210 DATA 213,106,119,126,142,15
9,169,189
220 DATA 255
```



```
10  RESTORE :T = 80
20  FOR I = 0 TO 22: READ A: PO
KE 800 + I,A: NEXT
30  DATA  160,0,174,133,3,238,4
8,192,136,208,5,206,132,3,240,6
,202,208,245,76,34,3,96
40  READ D
50  IF D = 255 THEN 10
60  POKE 900,T: POKE 901,D: CAL
L 800
70  GOTO 40
100  DATA  47,47,40,37,31,31,28
,31
110  DATA  47,23,26,28,31,35,37
,42
120  DATA  47,47,40,37,31,31,28
,31
130  DATA  47,23,26,28,31,35,37
,42
140  DATA  35,35,29,28,23,23,21
,23
150  DATA  35,23,26,28,31,35,37
,42
160  DATA   47,47,40,37,31,31,2
8,31
170  DATA   47,23,26,28,31,35,3
7,42
180  DATA  31,31,25,23,21,21,23
,25
190  DATA  35,35,29,28,23,23,29
,28
200  DATA   47,47,40,37,31,31,2
8,31
210  DATA   47,23,26,28,31,35,3
7,42
220  DATA  255
```



```
10 T=3
20 RESTORE
30 READ D
50 IF D=255 THEN 20
60 SOUND D,T
70 GOTO 30
100 DATA 175,175,189,193,204,20
4,210,204
110 DATA 175,218,213,210,204,19
7,193,185
120 DATA 175,175,189,193,204,20
4,210,204
130 DATA 175,218,213,210,204,19
7,193,185
140 DATA 197,197,207,210,218,21
8,223,218
```

```
150 DATA 197,218,213,210,204,19
7,193,185
160 DATA 175,175,189,193,204,20
4,210,204
170 DATA 175,218,213,210,204,19
7,193,185
180 DATA 204,204,216,218,223,22
3,218,216
190 DATA 197,197,207,210,218,21
8,207,210
200 DATA 175,175,189,193,204,20
4,210,204
210 DATA 175,218,213,210,204,19
7,193,185
220 DATA 255
```

Se quiserem, os usuários do TK-2000 poderão modificar este e os próximos programas a fim de usar o comando **SOUND**. Mas isto não é necessário: tal como estão, os programas funcionam tão bem em seu micro quanto no Apple.

Em geral, o MSX requer o dobro dos dados utilizados pelos outros micros, já que emprega dois bytes para definir a nota musical — o byte mais significativo e o menos significativo. A melodia que escolhemos, no entanto, é tão simples que o byte mais significativo de todas as suas notas é igual a 0, o que nos permite especificar só os bytes menos significativos.

A variável **T** estabelece um fator de tempo para a velocidade da música. A linha 20 acerta o apontador das notas no início dos dados. O laço **FOR ... NEXT** entre as linhas 30 e 70 lê os valores das notas — os números nas linhas **DATA** —, colocando-os dentro do comando musical apropriado na linha 60. A linha 50 detecta o fim da melodia, marcado por um valor arbitrário — 255.

Se quisermos introduzir pausas no meio da melodia, podemos definir um outro valor arbitrário, 254 por exemplo, e incluir um teste para esse valor na linha 40. Ao surgir uma pausa no meio dos dados, a linha 40 desviaria o programa para uma outra linha. Esta produziria um certo atraso, voltando depois para a linha 30 a fim de continuar a execução da música.

Tal como o programa está, as notas são fornecidas aos comandos musicais do BASIC e emitidas conforme vão sendo lidas nas linhas **DATA**.

Embora esse processo funcione a contento, ele exige um número muito grande de dados, mesmo para uma melodia curta. Além de dificultar a digitação e tomar espaço na memória, o excesso de dados traz ainda um outro inconveniente: enquanto a música está sendo lida e processada, o computador não pode se ocupar com outra tarefa.

Alguns micros, como o MSX, solucionam parcialmente esse problema com um *buffer* musical, capaz de guardar algumas notas. Se houver espaço no buffer para o som que está sendo emitido, o computador é liberado para outras atividades. Porém, havendo mais notas a processar, o micro fica novamente comprometido. Seja como for, o buffer não alivia o trabalho de digitação das linhas **DATA** nem reduz o espaço por elas ocupado na memória.

Para contornar efetivamente esses problemas, o usuário só tem uma alternativa: encontrar um meio de comprimir os dados, tornando mais fácil seu processamento e economizando espaço.

MÃOS À OBRA

A compressão de dados é viabilizada pela existência, em uma melodia, de padrões que se repetem. Quanto mais padrões pudermos reconhecer, maior será o grau de compressão.

Para identificar esses padrões de dados, convém, em primeiro lugar, ouvir ou tocar a melodia em um instrumento, tentando distinguir as passagens que se repetem. Depois, escreva a música em um papel, preocupando-se apenas com a freqüência das notas — ignore pautas, claves, compassos etc.

Não há dificuldade em escrever, em



seqüência, as letras equivalentes às notas, se estas têm sempre a mesma duração. Mas, e se as notas tiverem duração variada? Nesse caso, o trabalho torna-se um pouco mais complicado, pois precisaremos levar em conta, além da tonalidade, o tempo de cada uma — o que duplicará o volume de dados.

Outra saída consiste em repetir as notas de maior duração. Se, por exemplo, uma nota dura duas batidas, ela será registrada duas vezes. Esta foi a alternativa que utilizamos ao transcrever a melodia do programa anterior. Obtivemos a seguinte tabela:

TABELA 1

G	G	A#	B	D	D	E	D	G
G2	F	E	D	C	B	A		
G	G	A#	B	D	D	E	D	G
G2	F	E	D	C	B	A		
C	C	D#	E	G2	G2	A2	G2	C
G2	F	E	D	C	B	A		
G	G	A#	B	D	D	E	D	G
G2	F	E	D	C	B	A		
D	D	F#	G2	A2	A2	G2	F#	
C	C	D#	E	G2	G2	D#	E	
G	G	A#	B	D	D	E	D	G
G2	F	E	D	C	B	A		

Se você examinar a tabela 1 com atenção, verá que a melodia se compõe de apenas cinco seqüências, ou minimelodias — identificadas como T1, T2, etc., como mostra a tabela 2 —, em sua maioria tocadas repetidas vezes.

TABELA 2

T1 =	G	G	A#	B	D	D	E	D	G
T2 =	G2	F	E	D	C	B	A		
T3 =	C	C	D#	E	G2	G2	A2	G2	C
T4 =	D	D	F#	G2	A2	A2	G2	F#	
T5 =	C	C	D#	E	G2	G2	D#	E	

Podemos agora utilizar uma técnica de compressão: as minimelodias não serão digitadas sempre que aparecem na música completa, mas uma única vez, junto a códigos que especificam a seqüência em que devem ser tocadas. Como costuma ocorrer quando comprimimos dados, o programa que lê a melodia assim digitada é maior que a anterior:

S

```
10 LET C=0: LET T=.2
20 RESTORE 100
30 FOR N=1 TO C+1: READ P:
NEXT N
40 IF P=0 THEN  GOTO 10
50 RESTORE P
60 READ N
70 IF N>=255 THEN  LET C=C+1:
 GOTO 20
80 SOUND T,N
90 GOTO 60
100 DATA 110,120,110,120,130,
120,110,120,140,150,110,120,0
110 DATA 12,12,15,16,19,19,21,
19,12,255
120 DATA 24,22,21,19,17,16,14,
255
130 DATA 17,17,20,21,24,24,26,
24,17,255
140 DATA 19,19,23,24,26,26,24,
23,255
150 DATA 17,17,20,21,24,24,20,
21,255
```

MSX

```
1 SOUND 7,56:SOUND 8,15
2 SOUND 1,0
10 C=0:T=120
20 RESTORE
27 FORZ=1TOC+1:READP:NEXT
28 IFP=0THEN10
29 RESTORE:FORW=1TOP:READWW:NEX
T
30 READK:IFK=255THENC=C+1:GOTO2
0
50 SOUND 0,K
60 FORZ=1TOT:NEXT
70 GOTO30
100 DATA 13,23,13,23,31,23,13,2
3,41,50,13,23,0
110 DATA 213,213,179,169,142,14
2,126,142,213,255
120 DATA 106,119,126,142,159,16
9,189,255
130 DATA 159,159,134,126,106,10
6,94,106,159,255
140 DATA 142,142,112,106,94,94,
106,112,255
150 DATA 159,159,134,126,106,10
6,134,126,255
10013 ,23,13,23,0
```

```
1  FOR I = 0 TO 22: READ A: POK
E 800 + I,A: NEXT
2  DATA   160,0,174,133,3,238,4
8,192,136,208,5,206,132,3,240,6
,202,208,245,76,34,3,96
10 C = 0:T = 100
20  RESTORE
27  FOR Z = 1 TO C + 24: READ P
: NEXT
28  IF P = 0 THEN 10
29  RESTORE : FOR W = 1 TO P +
23: READ WW: NEXT
30  READ K: IF K = 255 THEN C =
 C + 1: GOTO 20
40  POKE 900,T: POKE 901,K
50  CALL 800
70  GOTO 30
100  DATA  13,23,13,23,31,23,13
,23,41,50,13,23,0
110  DATA   47,47,40,37,31,31,2
8,31,47,255
120  DATA   23,26,28,31,35,37,4
2,255
130  DATA    35,35,29,28,23,23,
21,23,35,255
140  DATA   31,31,25,23,21,21,2
3,25,255
150  DATA   35,35,29,28,23,23,2
9,28,255
220  DATA  255
```

T

```
1 DIM A(5,1):FOR K=1 TO 13:READ
 P:NEXT:GOTO 3
2 READ P:IF P<>255 THEN 2
3 N=N+1:A(N,0)=PEEK(51):A(N,1)=
PEEK(52):IF N<5 THEN 2
10 C=0:T=3
20 RESTORE
30 FOR N=1 TO C+1:READ P:NEXT
40 IF P=0 THEN 10
50 POKE 51,A(P,0):POKE 52,A(P,1
)
60 READ N
70 IF N=255 THEN C=C+1:GOTO 20
80 SOUND N,T
90 GOTO 60
100 DATA 1,2,1,2,3,2,1,2,4,5,1,
2,0
110 DATA 175,175,189,193,204,20
4,210,204,175,255
120 DATA 218,213,210,204,197,19
3,185,255
130 DATA 197,197,207,210,218,21
8,223,218,197,255
140 DATA 204,204,216,218,223,22
3,218,216,255
150 DATA 197,197,207,210,218,21
8,207,210,255
```

Note que o número de dados necessários para a execução da melodia foi muito reduzido — no Spectrum, por exemplo, diminuiu de 97 para 59 bytes. Rodando o programa, você verá que a música é exatamente a mesma do programa anterior. Os usuários do Spectrum vão estranhar o andamento da melodia, mas logo terão uma explicação.

Os dados que definem as minimelodias estão nas linhas 110 a 150 e a seqüência principal — isto é, a ordem em que as minimelodias são executadas — fica na linha 100. O laço na linha 30 acerta o valor de **P** (variável de trabalho utilizada na leitura das linhas **DATA**) de acordo com a seqüência principal, definindo a minimelodia a ser tocada. Na realidade, a seqüência princi-

pal é uma lista de números — ou de valores que, combinados com uma constante, resultam em números de linha — correspondentes às linhas onde se encontram as minimelodias.

Identificada a minimelodia a ser executada, a linha 50 calcula o número da linha **DATA** correspondente. Só o Spectrum permite o uso da instrução **RESTORE** durante a leitura do bloco de dados. O Apple e o MSX empregam laços **FOR...NEXT** e o TRS-Color tem os pontos de entrada gravados na linha 3.

A linha 10 utiliza a variável **C** para a contagem do número de minimelodias já tocadas. A duração de cada nota é controlada por **T**, que também determina o andamento da música. A linha 40 (28, no Apple e no MSX) verifica a última minimelodia, que é marcada com 0 na linha 100. O número 255 assinala o final de cada minimelodia. Ao encontrá-lo, o computador volta à seqüência principal para saber qual será a próxima minimelodia.

Existem várias alternativas para a divisão de uma música em seqüência principal e minimelodias. De um modo geral, quanto menores as minimelodias, maior é a seqüência principal. Devemos buscar um equilíbrio entre esses dois elementos, a fim de evitar que minimelodias muito curtas exijam uma seqüência principal tão longa que anule as vantagens da compressão.

DIVIDINDO PARA VENCER

A segunda técnica de compressão é ainda mais eficiente. Ela parte do princípio de que, embora os micros possam tocar um grande número de notas — geralmente 256 —, poucas delas são de fato utilizadas durante a execução de uma melodia. Assim, não nos interessa — e não é econômico — empregar um sistema planejado para aceitar um grande número de notas.

Cada posição de memória em seu microcomputador de oito bits é capaz de armazenar um número inteiro entre 0 e 255. Restringindo a quantidade de notas a serem armazenadas, será possível colocar duas delas em cada posição — uma em cada quatro bits. Por exemplo, o número binário 10100010, de oito bits, pode ser interpretado como um só valor decimal — 162 — ou como dois valores — 10 (1010 em binário) e 2 (0010 em binário).

Dividir por dois o espaço disponível diminui drasticamente o intervalo de valores que podemos armazenar: 0 a 15. Contudo, muitas vezes isso é suficiente, pois o número de notas diferentes em uma melodia simples raramente ultrapassa dezesseis.

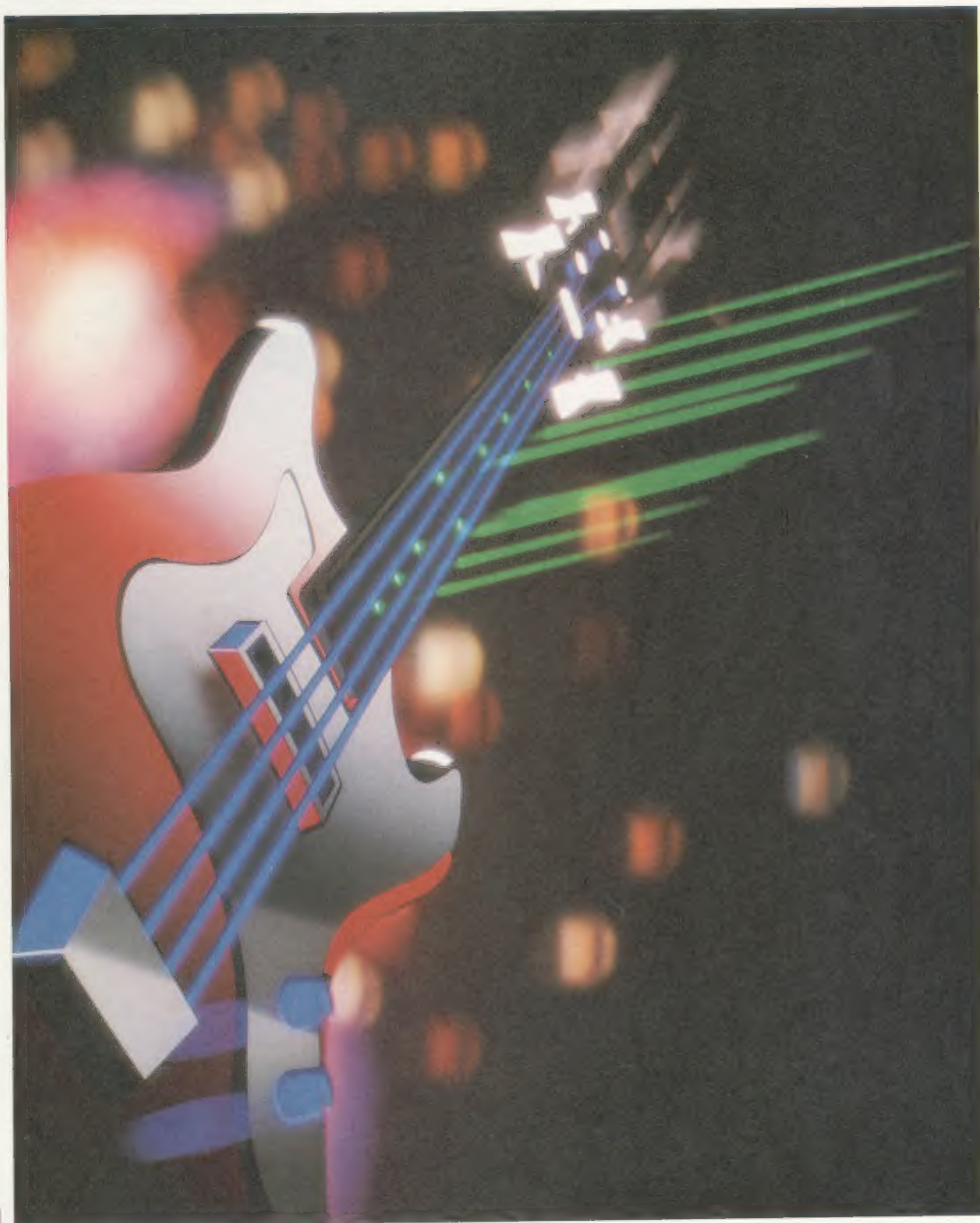
Para utilizar essa técnica de compressão, precisamos reduzir o número de notas a quinze, deixando o 16º valor como um código de controle. Já comentamos que, quanto mais comprimidos os dados, maior é o programa necessário para interpretá-los. Aqui, o programa deve decifrar cada nota abreviada que vai lendo nas linhas **DATA**.

O usuário, por sua vez, terá o trabalho de selecionar as notas que serão usadas na melodia, classificando-as em ordem crescente, conforme a freqüência, a partir da nota mais baixa da primeira oitava — nesse caso, G1. Em seguida, precisará calcular os números codificados que aparecerão nas linhas **DATA**. Nossa melodia inclui doze notas: G, A, A#, B, C, D, D#, E, F, F#, G2 e A2.

Digite o terceiro programa e veja o sistema em ação.

```
12 GOSUB 1000: LET T=.15: LET
 NT=130: LET MS=170: LET MT=
 210: GOSUB 300
 90 STOP
100 DATA 12,14,15,16,17,19,20,
21,22,23,24,26,0,0,0,0
200 DATA 1,2,1,2,3,2,1,2,4,5,1
,2,0
210 DATA 0,35,85,117,15,255
220 DATA 168,117,67,31,255
230 DATA 68,103,170,186,79,255
240 DATA 85,154,187,169,255
250 DATA 68,103,170,103,255
310 RESTORE NT
320 FOR N=23410 TO 23425
330 READ X: POKE N,X: NEXT N
345 LET NM=0
350 RESTORE MS: LET HL=23426
360 READ X
365 IF X>NM THEN  LET NM=X
370 IF X=0 THEN  GOTO 400
380 POKE HL,X: LET HL=HL+1:
GOTO 360
400 POKE HL,X: LET HL=HL+1
401 LET X=HL: GOSUB 600
402 POKE 23403,LSB
403 POKE 23404,MSB
430 RESTORE MT
440 FOR N=1 TO NM
450 READ X
460 IF X=255 THEN  GOTO 500
470 POKE HL,X: LET HL=HL+1:
GOTO 450
500 POKE HL,X: LET HL=HL+1
510 NEXT N
511 RAND USR 23371
512 POKE 23409,0
530 LET X=USR 23296
540 IF X=255 THEN  RETURN
550 SOUND T,X: GOTO 530
600 LET MSB=INT (X/256)
610 LET LSB=X-(MSB*256):
RETURN
1000 RESTORE 2000: LET TO=0: LE
T L=2000
1030 FOR N=23296 TO 23296+111 S
TEP 8
1040 FOR K=0 TO 7: READ A: LET
TO=TO+A: POKE K+N,A: NEXT K
1050 READ A: IF A<>TO THEN  GOT
O 1080
1060 LET L=L+10: LET TO=0: NEXT
 N
1065 RESTORE : RETURN
1080 PRINT "ERRO DE DADOS NA LI
NHA ";L: STOP
2000 DATA 42,109,91,235,42,111,
91,58,779
2010 DATA 113,91,70,183,202,24,
91,175,949
2020 DATA 8,35,62,15,160,195,36
,91,602
2030 DATA 61,1,8,203,56,203,56,
203,791
2040 DATA 56,203,56,120,254,15,
202,65,971
2050 DATA 91,34,111,91,235,34,1
09,91,796
2060 DATA 8,50,113,91,33,114,91
,22,522
2070 DATA 0,8,95,25,126,6,0,79,
339
2080 DATA 201,26,19,183,194,81,
91,1,796
2090 DATA 255,0,201,17,130,91,1
95,65,954
2100 DATA 91,71,42,107,91,43,62
,255,762
2110 DATA 16,5,35,175,195,10,91
,35,562
2120 DATA 190,194,103,91,195,88
,91,35,987
2130 DATA 195,96,91,0,0,0,0,0,3
82
```

```
10 SOUND 7,56:SOUND 8,15
20 SOUND 1,0
23 DIM X(16):RESTORE:FOR N=1 TO
 16:READ X(N):NEXT
25 C=0:T=100
26 RESTORE1000
27 FORZ=1TOC+1:READP:NEXT
28 IFP=0THEN25
29 RESTORE1010:FORW=1TOP+2:READ
WW:NEXT
50 READN:SS=N
60 N=INT(N/16)
70 IFN=15THENC=C+1:GOTO26
80 GOSUB130
90 N=SS:N=15ANDN
100 IFN=15THENC=C+1:GOTO26
110 GOSUB130
120 GOTO50
130 SOUND0,X(N+1)
140 FORZ=1TOT:NEXT
150 RETURN
450 DATA 213,189,179,169,159,14
2,134,126,119,112,106,94,0,0,0,
0
1000 DATA 5,10,5,10,15,10
1010 DATA 5,10,20,25,5,10,0
1110 DATA 0,35,85,117,15
1120 DATA 168,117,67,31,255
```



```
1130 DATA 68,103,170,186,79
1140 DATA 85,154,187,169,255
1150 DATA 68,103,170,103,255
```



```
1  FOR I = 0 TO 22: READ A: POK
E 800 + I,A: NEXT
2  DATA   160,0,174,133,3,238,4
8,192,136,208,5,206,132,3,240,6
,202,208,245,76,34,3,96
23  DIM X(16): FOR N = 1 TO 16:
 READ X(N): NEXT
25 C = 0:T = 100
26  RESTORE
27  FOR Z = 1 TO C + 40: READ P
: NEXT
28  IF P = 0 THEN 25
29  RESTORE : FOR W = 1 TO P +
47: READ WW: NEXT
50  READ N:SS = N
60 N =  INT (N / 16)
70  IF N = 15 THEN C = C + 1: G
OTO 26
80  GOSUB 130
90 N = SS - N * 16
100  IF N = 15 THEN C = C + 1:
GOTO 26
110  GOSUB 130
120  GOTO 50
130  POKE 900,T: POKE 901,X(N +
1)
140  CALL 800: RETURN
450  DATA  47,42,40,37,35,31,29
,28,26,25,23,21,0,0,0,0
1000  DATA  5,10,5,10,15,10
1010  DATA  5,10,20,25,5,10,0
1110  DATA  0,35,85,117,15
1120  DATA  168,117,67,31,255
1130  DATA  68,103,170,186,79
1140  DATA  85,154,187,169,255
1150  DATA  68,103,170,103,255
```



```
10 DIM A(5,1),X(16):FOR K=1 TO
16:READ X(K):NEXT:A(0,0)=PEEK(5
1):A(0,1)=PEEK(52)
15 FOR K=1 TO 13:READ P:NEXT:GO
TO 30
20 FOR K=1 TO 5:READ P:NEXT
30 N=N+1:A(N,0)=PEEK(51):A(N,1)
=PEEK(52):IF N<5 THEN 20
40 C=0:T=3
50 POKE 51,A(0,0):POKE 52,A(0,1
):FOR N=1 TO C+1:READ P:NEXT:IF
 P=0 THEN 40
60 POKE 51,A(P,0):POKE 52,A(P,1
)
65 READ N:S=N
70 N=INT(N/16)
75 IF N=15 THEN C=C+1:GOTO 50
80 GOSUB 130
90 N=S:N=15 AND N
100 IF N=15 THEN C=C+1:GOTO 50
110 GOSUB 130
120 GOTO 65
130 SOUND X(N+1),T:RETURN
450 DATA 175,185,189,193,197,20
4,207,210,213,216,218,223,0,0,0
,0
1000 DATA 1,2,1,2,3,2
1010 DATA 1,2,4,5,1,2,0
1110 DATA 0,35,85,117,15
1120 DATA 168,117,67,31,255
1130 DATA 68,103,170,186,79
1140 DATA 85,154,187,169,255
1150 DATA 68,103,170,103,255
```

Os valores das frequências dessas notas estão codificados na linha 450 (100 no Spectrum). Essa linha deve conter dezesseis valores. No MSX, poderíamos precisar de 32 valores para outras melodias, já que ele requer dois parâmetros para um valor de nota.

Em todas as versões, codificamos só doze notas; assim, quatro espaços são preenchidos com 0. Os quatro dígitos binários (chamados *nybble*) que formam cada nota das minimelodias estão codificadas nas linhas 1000 e 1010 (200 e 210 no Spectrum). Para entender como se calculam esses valores, tomemos como exemplo a minimelodia T1. Veja os números da seqüência principal na linha 450 (100 no Spectrum). Chamemos a primeira nota dessa linha de 0, a segunda de 1 e assim por diante, até a última nota, que será 15. Se escrevermos a minimelodia T1 com esses códigos, teremos T1 = 0, 0, 2, 3, 5, 5, 7, 5, 0, 15. O número 15, no caso, indica o final da minimelodia, e não uma nota. Cada um desses valores cabe em um nybble; assim, precisamos combiná-los em pares para obter os bytes.

O primeiro par de nybbles é 0 e 0, valores que correspondem em binário, a 0000 e 0000. Combinados em um byte, resultam no decimal 0. O primeiro valor da linha **DATA** 1110 (210 no Spectrum), que equivale a T1, é, portanto, 0. O próximo par é 2 e 3 — em binário, 0010 e 0011. Combinados, esses valores resultam em 00100011, ou 35 em decimal. Este é o segundo valor da linha 1110 (ou 210). Procedendo dessa maneira com a minimelodia seguinte, teremos T2 = 10, 9, 7, 5, 4, 3, 1, 15. Combinados, os valores 10 e 9 — 1010 e 1001 em binário — resultam 168 em decimal. Este é o primeiro valor de T2 na linha 1120 (220 no Spectrum).

Utilizamos esse método para calcular os valores correspondentes às minimelodias, armazenados nas linhas 1110 a 1150 (210 a 250 no Spectrum). Como no programa anterior, a seqüência principal (linhas 1000 e 1010, ou 200 no Spectrum) aponta para as linhas **DATA** onde estão as minimelodias, na ordem em que devem ser tocadas.

O PROCESSO INVERSO

Outra parte importante do programa é a que se encarrega de extrair os nybbles dos números decimais lidos nas linhas **DATA**. O Spectrum faz isso colocando o valor em uma variável (linha 401) e chamando, em seguida, uma sub-rotina (linhas 600 e 610) que separa os dois nybbles do byte. Devidamente processados, esses valores são utilizados para emitir duas notas.

Nos outros micros, as linhas 50 a 100 cuidam da decodificação. Primeiro, o byte em exame, **N**, é armazenado na linha 50. A linha 60 extrai o nybble mais significativo e a linha 70 verifica o código de final de melodia. A linha 80 chama a sub-rotina que executa a nota correspondente e a linha 90 extrai o nybble menos significativo, usando a operação lógica **AND** (menos no Apple e no TK-2000, cuja função **AND** não permite o cálculo de valores diferentes de 0 e 1). A nota é então emitida. Note que o byte em exame é preservado, sem o que não seria possível recuperar o segundo nybble.

Ao executar esse programa, os usuários do Spectrum observarão que os problemas de andamento causados pela compressão já não ocorrem. O defeito era provocado por um tempo extra de processamento, necessário ao realinhamento dos ponteiros, ou seja, ao uso de **RESTORE**. O programa reduziu esse tempo de processamento por meio de uma rotina em código, armazenada nas linhas 1000 a 2130. Esta é a razão das diferenças de tamanho e numeração entre o Spectrum e os demais micros.

Para alterar a velocidade de execução, altere o valor de **T**. Ele fica na linha 12, no Spectrum, e na linha 40, nos outros computadores. Observe, porém, que o intervalo de tempo entre o final de uma minimelodia e o início da seguinte torna-se mais acentuado à medida que **T** diminui.

MICRO DICAS

MAIS MELODIAS

Como exercício, tente codificar outras melodias, utilizando o último programa deste artigo. Identifique as minimelodias e as seqüências principais, calcule os valores correspondentes e coloque-os em linhas **DATA**. Será preciso redefinir as matrizes — **X( )** — da seqüência principal para cada nova melodia, o que pode exigir a renumeração do programa

AS CINCO VIDAS DE WILLIE

A morte vem para todos — mas não tão insistentemente como para Willie. Ele morre cinco vezes em cada jogo... e está sempre pronto a reiniciar a aventura na hora que você quiser!

Mesmo que você seja um exímio jogador, mais cedo ou mais tarde a morte levará Willie. Mas não se afobe. Ele tem cinco vidas — nem tantas como um gato; bem mais, porem, do que os pobres mortais. Quando a fatalidade atingir Willie definitivamente, você deverá enterrá-lo, imprimir o placar final e ajustar todas as variáveis.

Os sons emitidos pelo Spectrum são muito simples, acompanhando a execuçao dos programas. O TRS-Color pode produzir sons sofisticados. Mas, como para isso é necessário utilizar o processador, não se pode tocar um som e executar o jogo simultaneamente.



A rotina apresentada a seguir promove a morte de Willie e verifica se ele pode ser ressuscitado.

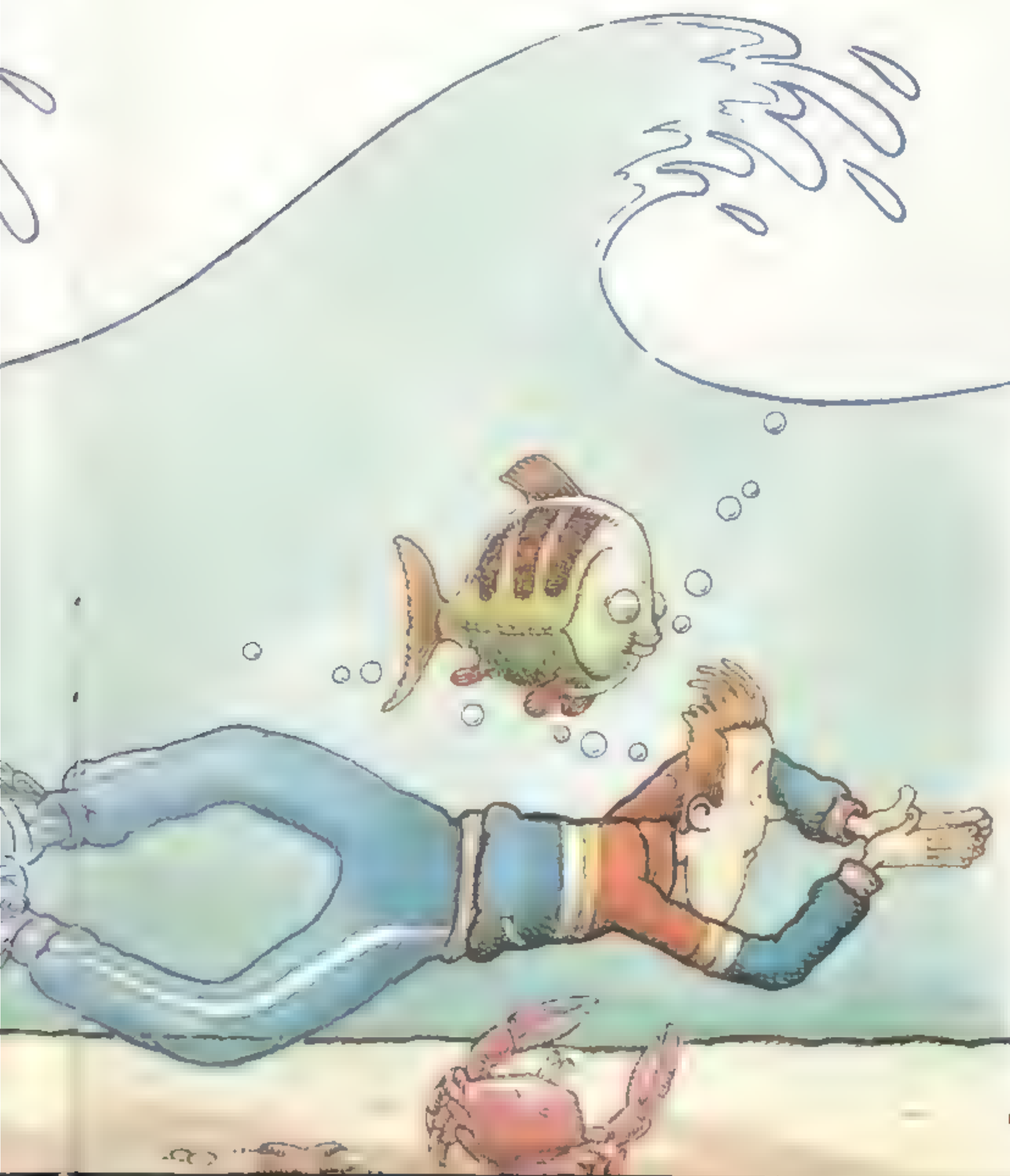
```
10 REM org 59652
20 REM die ld de,196
30 REM ld hl,1086
40 REM call 949
50 REM ld de,131
60 REM ld hl,1646
70 REM call 949
80 REM dead ld hl,(57332)
90 REM ld bc,15616
100 REM ld a,45
110 REM call 58217
120 REM ld de,32
130 REM add hl,de
140 REM ld (57332),hl
150 REM ld bc,57000
160 REM ld a,40
170 REM ld de,258
180 REM call 58970
190 REM ld de,30
200 REM ld hl,878
210 REM ld a,r
```

ão

- NOTAS FÚNEBRES
- DESCIDA PARA O INFERNO
- PERDENDO VIDAS
- A ÚLTIMA MORTE DE WILLIE
- FUNERAL
- IMPRESSÃO DO PLACAR
- SONS FINAIS

```
220 REM ld l,a
230 REM call 949
240 REM ld hl,(57332)
250 REM ld de,704
260 REM sbc hl,de
270 REM jr c,dead
280 REM ld a,9
290 REM ld (60005),a
300 REM ld a,(57343)
310 REM dec a
320 REM ld (57343),a
330 REM jp nz,58606
340 REM ld hl,330
350 REM ld a,142
360 REM ld b,11
370 REM ld ix,57992
380 REM call me
390 REM call 58939
400 REM ld de,261
410 REM ld hl,1646
420 REM call 949
430 REM ld de,392
440 REM ld hl,1086
450 REM call 949
460 REM ld de,261
470 REM ld hl,1646
480 REM call 949
490 REM ld a,50
500 REM ld (58732),a
510 REM jp 58576
```

Se Willie se atogou no mar, caiu num buraco, foi picado por uma cobra ou atingido por uma pedra, os sinos tocam, anunciando sua morte. Para que soem as duas badaladas fúnebres, a rotina **BEEPER** é chamada em 949 duas vezes — em cada uma delas, com parâmetros diferentes nos pares HL e DE.



DESCIDA PARA O INFERNO

Quando Willie morre, ele desce para o inferno, que se situa no fundo da tela. Para criar esse efeito, começamos por apagar sua cabeça — se não o fizermos, teremos uma coluna de cabeças descendo até o fundo da tela!

Assim, a posição de Willie na tela — que aponta para a posição de sua cabeça — é carregada da variável em 57332 e 57333 para o par HL. BC é carregado com o endereço de um caractere limpo de céu do topo da tela. A é ajustado para ciano sobre ciano. Finalmente, a rotina **print** em 58217 é chamada, apagando a cabeça de Willie.

O valor 32 é então somado ao apontador de tela em HL, através do par DE, o que o faz se mover para a linha de baixo da tela. O resultado é colocado de volta em 57332 e 57333 para ajustar também a variável.

BC é carregado com o endereço inicial da figura de Willie com as pernas juntas. A é ajustado com o código da cor de Willie, azul sobre ciano. DE é carregado com 258, especificando um bloco de um por dois — o conteúdo do registro D define a largura do bloco, e o conteúdo do registro E, sua profundidade: 1 x 256 + 2 = 258. A rotina de impressão de bloco em 58970, **blk**, é chamada, exibindo na tela a figura de Willie com as pernas juntas, uma posição abaixo da que ele ocupava antes.

Enquanto Willie desce, ouve-se uma música "celestial". Mais uma vez, os pares de registros HL e DE são carregados com parâmetros que a rotina **BEEPER** utilizará, quando for chama-

da em 949. Desta vez, porém, em vez de um som constante, o programa precisa de um som intermitente. Para isso, carrega-se o conteúdo do registrador R na parte menos significativa de HL, que é o registro L. A operação é realizada via acumulador porque não existe nenhuma instrução **ld l,r**.

Usamos o registro R para manter a memória dinâmica. Este é um recurso do hardware com o qual você não precisa se preocupar. Para nossos fins, basta saber que o valor desse registro é variável. Assim, cada vez que esta parte da rotina for acessada irá produzir um som diferente.

SEIS PÉS DE PROFUNDIDADE

A posição da cabeça de Willie é carregada da variável em 57332 e 57333 para o par HL; 704 é carregado em DE e subtraído do conteúdo de HL. O número 704 — 32 × 22 — indica o começo da penúltima linha. Como Willie tem dois caracteres de altura, essa subtração colocará o valor 1 na baliza *carry* até que os pés de Willie tenham tocado o fundo da tela.

Enquanto a baliza carry contiver 1, a instrução **jr c, dead** manda o processador de volta ao início da rotina de enterro do personagem. Willie será afundado mais uma posição e os sinos voltarão a tocar — num tom diferente. Quando os pés de Willie tocarem o inferno, ele estará completamente enterrado. A baliza carry não será ajustada com 1 pela subtração, não haverá salto e o processador passará a executar a próxima instrução.

Tendo Willie chegado ao seu destino final, convém tocar uma canção confortadora. Assim, carrega-se A com 9 e a rotina **sound** em 60006 é chamada. Ela executa uma versão curta da melodia inicial do jogo. Para obter maiores detalhes sobre rotina da música, reveja o artigo da página 788.

Como nosso personagem perdeu uma vida, precisamos reajustar — ou melhor, decrementar — o número de vidas. O valor dessa variável é carregado de 57343 para o acumulador, decrementado e recolocado em 57343.

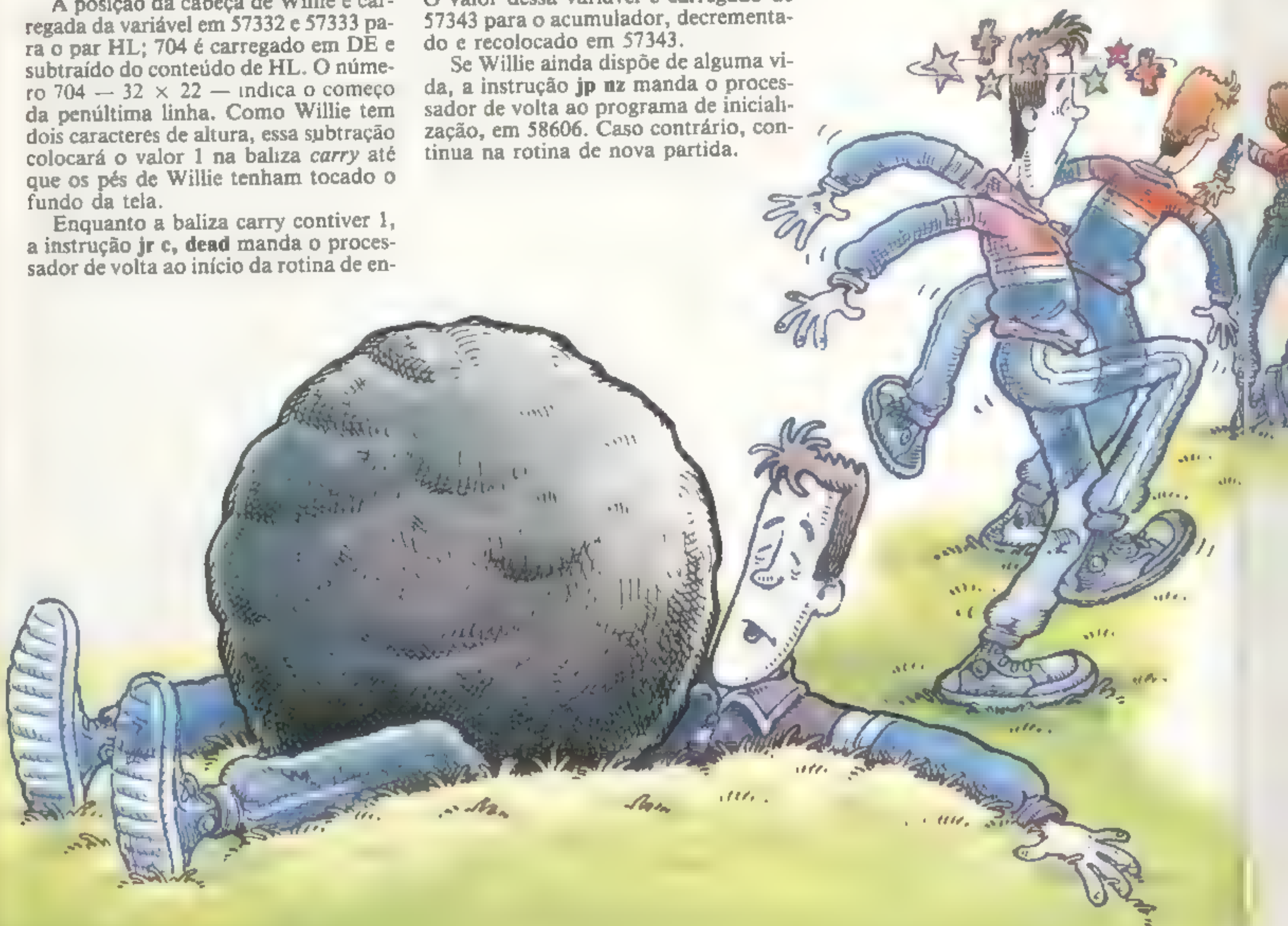
Se Willie ainda dispõe de alguma vida, a instrução **jp nz** manda o processador de volta ao programa de inicialização, em 58606. Caso contrário, continua na rotina de nova partida.

ACABARAM-SE AS VIDAS

Quando Willie tiver morrido pela quinta vez, o jogo acaba. Precisamos, então, imprimir a mensagem "GAME OVER!". Para isso, chamamos a rotina de mensagens, ou **me**, em 58155. Como sempre, os parâmetros devem ser colocados nos registros HL, A, B e IX.

A posição de impressão é carregada em HL. A cor é ajustada em A. B carrega o comprimento da mensagem e IX é usado como apontador de dados. Os dados para a mensagem estão em 57892.

A rotina que imprime o escore em 58939 é chamada para imprimir o placar. Em seguida, ouve-se o toque que marca o final do som no jogo. Para



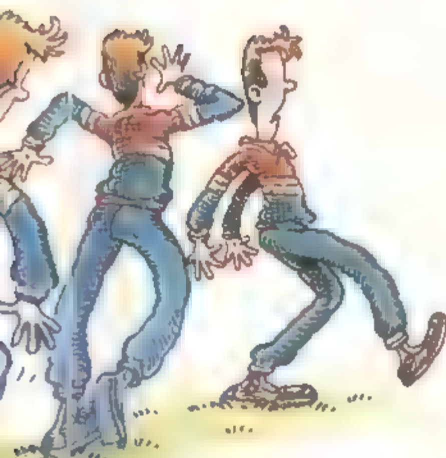
produzi-lo, a rotina **BEEPER** é chamada em 949 três vezes, com valores diferentes em HL e DE a cada vez.

Para reajustar o atraso inicial, carrega-se o acumulador com 50 e transfere-se o valor na posição 58732. Esse endereço ainda não foi mencionado: ele pertence à rotina final do jogo e é incumbido de controlar o atraso geral. O processador salta para 58578, ajustando-o de maneira que você possa dar reinício à aventura, se quiser.



A rotina a seguir coloca Willie em seu túmulo:

```
10   ORG 20560
20   DIE LDA #136
30   LDX #140
40   JSR SOUND
50   LDA #131
60   LDX #213
70   JSR SOUND
80   DI LDX 18249
90   LDU #1536
100   JSR CHARPR
110   LEAX 254,X
120   STX 18249
130   LDU #17774
140   JSR CHARPR
150   LEAX 254,X
160   JSR CHARPR
170   LDA #30
180   LDX #113
190   JSR SOUND
200   LDX 18249
210   CMPX #6912
220   BLO DI
230   DEC 18239
240   LBNE NLV
250   PB LDA #5
260   PAA LDX #65535
270   LEAX -1,X
280   BNE PAA
290   DECA
300   BNE PB
310   JSR CLS
320   LDA $FF22
330   ANDA #15
340   STA $FF22
350   STA $FFC2
360   STA $FFC4
370   STA $FFC6
380   LDY #$550B
390   LDX #$701
400   STX ,Y++
410   LDX #$D05
420   STX ,Y++
430   LDX #$200F
440   STX ,Y++
450   LDX #$1605
460   STX ,Y++
470   LDX #$1221
480   STX ,Y++
490   LDA #200
500   LDX #255
510   JSR SOUND
520   LDA #200
530   LDX #200
540   JSR SOUND
550   LDA #255
560   LDX #255
570   JSR SOUND
580   LDA #100
590   STA DLL+1
600   LBRA GBIN
610   SOUND EQU $5133
620   CHARPR EQU $4BCA
630   DLL EQU $51ED
640   GBIN EQU $4BE2
650   NLV EQU $4BF7
660   CLS EQU $4ACC
```



OS SINOS

Quando a rotina começa, ouve-se um sino bater duas vezes. A homenagem sonora ao finado Willie, porém, é promovida pela rotina **SOUND**, que será fornecida no próximo artigo de *Avalanche*.

Antes de testar o programa aqui apresentado, lembre-se de colocar um **RTS** no endereço inicial de **SOUND**, $5133. Caso contrário, haverá um erro.

SOUND requer dois parâmetros que especifiquem o tipo e a duração do som. Eles são utilizados pela rotina nos registradores A e X. Como as duas notas produzidas diferem uma da outra, a rotina **SOUND** trabalha com parâmetros distintos a cada chamada.

DESCENDO PARA O INFERNO

Willie descerá na tela em direção ao inferno. Sua figura anterior deve, assim, ser apagada.

X é carregado com o conteúdo de 18249, que aponta para a posição que o personagem ocupa na tela; U é carregado com 1536, o endereço do céu no topo da tela. Em seguida, a rotina **CHARPR** é chamada e imprime um bloco de céu, apagando a metade superior da figura de Willie. O conteúdo de X é então adicionado a 254 e recolocado em 18249, para que o apontador de Willie desça um caractere na tela.

U é carregado com 17774, o endereço inicial dos dados para a figura de Willie com as pernas juntas. Depois, o processador salta para a rotina **CHARPR**, que imprime a metade superior de Willie uma posição abaixo da última impressão. Novamente, X é incrementado com 254 e **CHARPR** é chamada para imprimir a metade inferior da figura.

Enquanto Willie está descendo na tela, ouve-se um grito de agonia — para isso, A e X são carregados de novo e o processador salta para **SOUND**.

Willie deve chegar ao fundo da tela. Verificamos se isso já ocorreu carregando X com o conteúdo de 18249 e comparando esse valor com 6912, o início de 28ª linha da tela. Se a posição de Willie em X for menor do que 6912, a instrução **BLO** manda o processador reiniciar o laço **DI** e move Willie um caractere para baixo. Caso contrário, o processador segue em frente, pois Willie chegou ao fundo da tela.

O conteúdo de 18249 — que armazena o número de vidas que restam a Willie — é então decrementado e **LBNE NLV** manda o processador para a rotina que dá nova vida a Willie e traz de volta a última tela. Uma instrução de desvio longo é usada neste ponto, porque a rotina **NLV** encontra-se numa parte bem anterior do programa — certamente mais do que 128 bytes, o limite para um desvio curto.

OUTRA PARTIDA

Se o conteúdo de 18249 foi reduzido a 0, Willie não tem mais nenhuma vida, o jogo terminou e o processador executa a instrução seguinte.

A é carregado com 5; X é carregado com 65535 e, depois, decrementado. O processador fica nesse laço até que X seja 0. Em seguida, A é decrementado e X volta a ser carregado com 65535. O processo se repete até que A também tenha sido reduzido a 0.

O processador fica nesse laço 5 x 65535 vezes — pausa suficiente para que o jogador medite sobre o trágico destino de Willie. Ao sair do laço de atraso, o processador salta para a rotina **CLS**, que limpa a tela.

O conteudo de $FF22 é então carregado em A, onde se executa a operação lógica **AND** com o valor 15. O resultado é armazenado nas posições de memória $FF22, $FFC2, $FFC4 e $FFC6.

Informamos, assim, ao VDG (gerador do sinal de vídeo) e ao SAM (multiplexador síncrono de endereços) que estamos voltando ao modo texto. O procedimento e o inverso do que utilizamos ao mudar para o modo gráfico.

Y é carregado com $50B, posição onde iremos imprimir a expressão "GAME OVER!". X, por sua vez, é carregado com $701, códigos de tela para as letras GA. Essas letras são colocadas na tela na posição apontada por Y, que é incrementado duas vezes

Em seguida, X é carregado com os códigos de tela para ME e as duas letras são impressas na nova posição apontada por Y, dois blocos à direita do início de GA. Por fim, os últimos caracteres — O, VE e R! — são carregados e impressos nas posições adequadas.

Um som anuncia o fim do jogo. Como esse som é formado por três notas, A e X são carregados e a rotina **SOUND** é chamada três vezes.

O numero 100 e carregado no acumulador e armazenado em $51EE. Com isso, ajustamos o atraso com o valor inicial. Quando o jogo for novamente executado, começará com a mesma velocidade.

Encerrando a execução da rotina, o processador faz um longo desvio para o início do programa em **GBIN**.

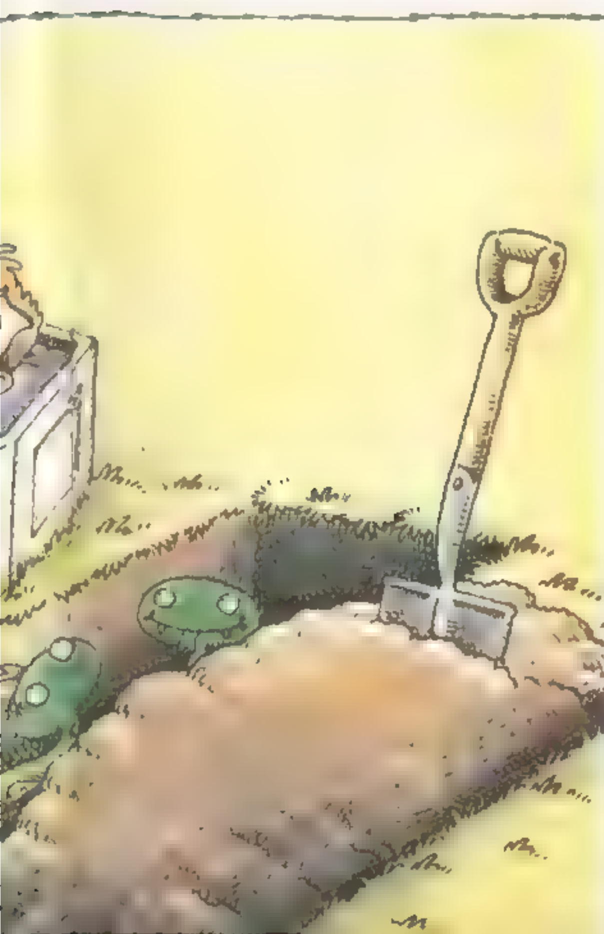


A rotina a seguir enterra Willie e verifica se pode ser ressuscitado:

```
10    org 55334
20    ld hl,(-5205)
30    ld de,(62407)
40    add hl,de
50    inc hl
60    ld a,255
70    push hl
80    call 77
90    pop hl
100    dec hl
110    dec hl
120    ld a,255
130    push hl
140    call 77
150    pop hl
160    ld de,32
170    add hl,de
180    ld a,255
190    push hl
200    call 77
210    pop hl
220    inc hl
230    inc hl
240    ld a,255
250    push hl
260    call 77
270    pop hl
280    inc hl
290    ld a,255
300    call 77
310    mo ld hl,(-5205)
320    push hl
330    ld de,(62407)
340    add hl,de
350    ld a,255
360    call 77
370    pop hl
380    ld de,32
390    add hl,de
400    ld (-5205),hl
410    ld de,(62407)
420    add hl,de
430    ld a,0
440    push hl
450    call 77
460    pop hl
470    ld de,32
480    add hl,de
490    ld a,1
500    call 77
510    ld b,255
520    atr ld a,255
530    dl dec a
540    jr nz,dl
550    djnz atr
560    ld hl,(-5205)
570    ld de,704
580    sbc hl,de
590    jr c,mo
600    ld a,(-5221)
610    dec a
620    ld (-5221),a
```

```
630    jp nz,53888
640    ld de,234
650    ld a,114
660    ld b,12
670    tt push bc
680    push de
690    push af
700    ld hl,(62407)
710    add hl,de
720    call 77
730    pop af
740    inc a
750    pop de
760    inc de
770    pop bc
780    djnz tt
790    ld b,30
800    tu ld a,255
810    du ld c,255
820    su dec c
830    jr nz,su
840    dec a
850    jr nz,du
860    djnz tu
870    call 54023
880    ld a,50
890    ld (54133),a
900    jp 53855
910    end
```

Essa rotina é executada quando nosso personagem morre — depois de se afogar no mar, cair num buraco, ser picado por uma cobra ou ser atingido e esmagado por uma pedra.



APAGANDO OS VESTÍGIOS

Antes de mandar Willie para o inferno, a rotina apaga seus vestígios na superfície. Se ele foi atingido por uma pedra, esta também sera apagada, evitando a deformação de figuras. Para isso, coloca-se o padrão do céu, código 255, em cinco posições ao redor de Willie. Examinando todas as situações de colisão, você verá que essas são as posições necessárias.

O par HL é carregado com a posição de Willie na tela, que está no apontador em −5205 e −5204; DE é carregado com o endereço inicial da TN da VRAM e somado em HL, o apontador na impressão. Em seguida, a rotina 77 da ROM é chamada cinco vezes, com o valor de HL apontando para cinco diferentes posições ao redor de Willie. Lembre-se de que a rotina 77 — utilizada no decorrer do jogo para imprimir um padrão na tela — coloca o valor de A no endereço da VRAM apontado por HL.

DESCIDA PARA O INFERNO

Willie chega até o fundo da tela, em sua descida ao inferno. Para criar esse efeito, começamos por apagar sua cabeça — se não o fizermos, veremos no vídeo uma coluna de cabeças!

A posição de Willie, que aponta para a sua cabeça, é carregada da variável em −5205 e −5204 para HL, par de registros guardado na pilha. O endereço inicial da TN na VRAM é carregado em DE e somado em HL. A rotina 77 coloca o padrão 255 nessa posição, apagando a cabeça de Willie. A posição é recuperada da pilha para HL, sendo somada a 32 através de DE e colocada de volta em −5205 e −5204. Assim, Willie afunda uma posição na tela.

Esse novo valor é somado com o endereço inicial da TN na VRAM através de DE. O acumulador é carregado com 0, o código do primeiro padrão da figura de Willie com as pernas juntas. O apontador é colocado na pilha e a rotina 77 é chamada. O apontador é recuperado da pilha e somado com 32, movendo-se para a posição de baixo. A é carregado com 1, o código do segundo padrão da figura de Willie. Chama-se a rotina 77 e Willie desce uma posição.

Em seguida, B e A são carregados com 255. O acumulador é decrementado até 0 no laço **dl**. Quando chega a esse valor, o processador, por meio da instrução **djnz atr**, decrementa B, reajusta A com 255 e repete o processo até que B também seja 0. Esse laço é executado 255 x 255 vezes e tem como único fim atrasar a descida de Willie, para que o jogador possa vê-la.

A posição de Willie é carregada da variável em −5205 e −5204 para HL, onde é subtraída de 704 através de DE. Esse valor indica o começo da penúltima linha na tela. Como Willie tem dois caracteres de altura, a subtração irá colocar o valor 1 na baliza carry até que seus pés tenham tocado o fundo da tela. Enquanto a baliza tiver esse valor, a instrução **jr c,mo** manda o processador de volta ao início da rotina do enterro, que afunda Willie mais uma posição. Quando seus pés tocarem o chão, ele estará completamente enterrado; a baliza carry não terá mais o valor 1 e não haverá o salto. O processador passa, então, a executar a próxima instrução.

O número de vidas precisa ser reajustado — ou melhor, decrementado, pois o pobre Willie perdeu uma vida. O número de vidas é carregado no acumulador em −5221, decrementado e recolocado no mesmo endereço. Se ainda restam vidas a Willie, o processador salta para o programa de inicialização, em 53888. Caso contrário, continua na rotina de nova partida.

ACABARAM-SE AS VIDAS

Quando Willie tiver consumido suas cinco vidas, o jogo acaba. Devemos, então, imprimir "NOVA PARTIDA", com os padrões previamente colocados na tabela de padrões da VRAM pela rotina fornecida no artigo da página 1001.

O par DE é carregado com 234, posição da tela onde a primeira letra será impressa. A contém o código de padrão da primeira letra; B, o número de padrões (doze, incluindo o espaço em branco). O laço **tt** é executado doze vezes, usando a rotina 77 para imprimir a mensagem no centro da tela. Note que este trabalho foi bem simplificado, porque os padrões para a mensagem já estavam na VRAM, em seqüência.

Depois, o laço formado por **tu**, **du** e **su** é executado 30 x 255 x 255 vezes, promovendo o atraso necessário para a mensagem ser lida na tela. A rotina que imprime o escore em 54023 é chamada para imprimir o placar final.

Para reajustar o atraso geral do jogo, carregamos o acumulador com 50 e transferimos esse valor para 54133 — endereço que pertence à rotina final e controla a velocidade do jogo.

O processador salta para 53855, o início da rotina principal, para que você tente de novo, se quiser.

OPERAÇÕES COM CADEIAS

A programação dos blocos básicos de um processador de textos em BASIC não é difícil. Aprenda aqui a remover espaços em branco e a converter maiúsculas em minúsculas e vice-versa.

A linguagem BASIC tem excelentes recursos para programação com variáveis literais (alfanuméricas). Diversas funções de manipulação de cadeias de caracteres (*string*) garantem a essa linguagem uma grande versatilidade. Vamos relembrar, resumidamente, o que faz cada uma dessas funções.

Funções gerais

LEN(X$) - Retorna o comprimento da cadeia **X$** (um número inteiro entre 0 e 255) Existe para todas as versões de BASIC. Nos microcomputadores ZX-81 e Spectrum não é necessário colocar **X$** entre parênteses.

ASC(X$) - Retorna o código ASCII (um número inteiro entre 0 e 255) do primeiro caractere da cadeia **X$**. No micro Sinclair ZX-81 denomina-se **CODE**.

CHR$(X) - Tem a função inversa de **ASC**, ou seja, retorna o caractere correspondente ao código ASCII indicado pela variável **X**.

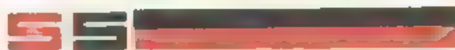
STRING$(X,Y) - Retorna uma cadeia com **X** caracteres iguais, com código ASCII estipulado por **Y**. Não existe para os microcomputadores das linhas Apple, Sinclair ZX-81 e Spectrum.

Funções de subcadeias

LEFT$ (X$,N) -Extrai os **N** primeiros caracteres da cadeia **X$**

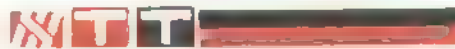
RIGHT$ (X$,N)**-Extrai os N últimos caracteres da cadeia **X$.

MID$ (X$,N,L) -Extrai **L** caracteres a partir da posição N da cadeia **X$**. Pode ser utilizado também no formato **MID$(X$,N)**, extraindo então todos os caracteres que existirem na cadeia **X$**, a partir da posição **N**.



Nos micros da linha Sinclair — ZX-81 e Spectrum — não são necessárias funções especiais para tratamento de subcadeias. Damos, abaixo, a equivalência entre a referência padronizada a subcadeias na linha Sinclair e as funções anteriores mencionadas.

X$(1 TO N)	⇔ **LEFT$(X$,N)**
X$(LEN X$-N TO+1)	⇔ **RIGHT$(X$,N)**
X$(N TO N+L-1)	⇔ **MID$(X$,N,L)**
X$(N TO)	⇔ **MID$(X$,N)**



INSTR (X$,Y$) - A função *instring* retorna a posição em que se está o caractere **Y$** (ou o primeiro caractere da cadeia **Y$**), dentro da cadeia **X$**.

Essa função pode ser simulada nos computadores onde não existe, por meio de um laço de repetição em que cada caractere em **X$** é extraído, sucessivamente, sendo comparado com **Y$**. Isso pode ser feito com a função **MID$(X$,I,1)**, onde **I** é o indicador do laço.

Funções de conversão



Números podem ser representados na forma de uma cadeia de dígitos, mais os sinais de ponto (.), mais (+), menos (-), e as letras E ou D (notação científica em precisão simples ou dupla, respectivamente. Esta última só existe nos micros das linhas TRS-80, TRS-Color e MSX). Para efetuar a conversão da representação numérica para cadeia, ou vice-versa, existem duas funções especiais em ASCII:

STR$(X) - Retorna a cadeia correspondente ao número **X**. Um espaço em branco é adicionado ao início da cadeia.

VAL (X$) - Efetua a operação inversa, ou seja, retorna o valor numérico correspondente à cadeia **X$**. Se **X$** não contiver um número, não ocorre erro: o valor que se retorna é igualado a 0.

Nem todas as versões do BASIC dispõem do conjunto completo de funções string. Além disso, faltam certas funções, necessárias para a programação avançada com cadeias (sobretudo em processamento de textos):

- eliminação de espaços em branco no começo ou no fim de uma cadeia;
- conversão de minúsculas para maiúsculas e vice-versa;
- extração de palavras de uma frase.

Neste artigo, examinaremos como resolver alguns dos problemas mencionados acima por meio de pequenas sub-rotinas, que podem ser acrescentadas a qualquer programa.

REMOÇÃO DE ESPAÇOS

Para muitas aplicações de processamento de cadeias, é preciso remover antes os espaços em branco existentes no começo ou no final de um string. A sub-rotina que apresentamos a seguir mostra como remover brancos à direita. Nela, a variável **A$** contém a cadeia que deve ser processada, e **B$**, o resultado da rotina:



```
1000 FOR I=LEN(A$) TO 1 STEP -1
1010 IF MID$(A$,I,1)<>" " THEN
1030
1020 NEXT I
1030 B$=LEFT$(A$,I):RETURN
```



```
1000 FOR I=LEN A$ TO 1 STEP -1
1010 IF A$(I TO I)<>" " THEN
GOTO 1030
1020 NEXT I
1030 LET B$=A$(TO I)
1040 RETURN
```

Para testar a rotina, acrescente este pequeno programa:



```
10 PRINT "ENTRE A CADEIA ";
20 INPUT A$
```

- MANIPULAÇÃO DE CADEIAS DE CARACTERES
- CONJUNTO DE FUNÇÕES STRING
- NOVAS ROTINAS DE PROCESSAMENTO DE TEXTOS
- ELIMINAÇÃO DE ESPAÇOS EM BRANCO
- CONVERSÃO PARA MAIÚSCULAS
- CONVERSÃO PARA MINUSCULAS

```
30 GOSUB 1000
40 PRINT B$
50 GOTO 10
```

A linha 1000 da sub-rotina percorre a cadeia de caracteres do fim ao começo, recuando um caractere por vez. A linha 1010 verifica se o caractere é um espaço em branco. Enquanto for, o laço se repete. A ocorrência do primeiro caractere não branco causa a interrupção do laço e a extração dos I caracteres à esquerda (linha 1030).

Ao testar o programa nos micros das linhas TRS-80, TRS-Color, MSX, Apple ou TK-2000, não se esqueça de digitar a cadeia entre aspas, pois, normalmente, o comando **INPUT** já realiza a função de retirar os espaços colocados à direita da mesma.

Nas versões do BASIC que permitem funções programadas (**DEF FN**), a rotina anterior pode ser substituída por algo bem mais compacto:

```
5 DEF FNTB$(A$)=LEFT$(A$,INSTR(
A$+"  "," ")-1)
```

Para fazer um teste, troque a linha 30 do programa de verificação por:

```
30 B$=FNTB$(A$)
```

Observe que a função só opera corretamente se não houver mais do que um espaço em branco entre as palavras de uma cadeia.

Para remover os espaços em branco à esquerda podemos usar a seguinte versão da rotina anterior:

```
1200 FOR I=1 TO LEN(A$)
1210 IF MID$(A$,I,1)<>" " THEN
1230
1220 NEXT I
1230 B$=MID$(A$,I):RETURN
```

```
1200 FOR I=1 TO LEN A$
1210 IF A$(I TO I)<>" " THEN
GOTO 1230
1220 NEXT I
1230 LET B$=A$(I TO)
1240 RETURN
```

Para testar a rotina, substitua a linha 30 do programa de verificação por **GOSUB 1200**.

Note que apenas a primeira e a última linhas da sub-rotina são diferentes, pois é necessário percorrer a cadeia do primeiro ao último caractere, até encontrar o primeiro não branco. Feito isso, eliminam-se todos os caracteres à direita.

MINÚSCULAS E MAIÚSCULAS

Outro recurso útil é a conversão de todas as letras minúsculas de uma cadeia em maiúsculas, ou vice-versa. Lembre-se de que o código ASCII de uma letra minúscula é igual ao código da maiúscula correpondente, mais o valor 32. O código de A, por exemplo, é 65, e o de a, 97. Portanto, para converter maiúsculas em minúsculas, precisamos apenas somar 32 ao código das letras — quando o código ASCII for menor do que 65, que é a letra A, ou maior do que 90, que é a letra Z, o caractere não deve ser convertido.

```
1300 B$=""
1305 FOR I=1 TO LEN(A$)
1310 C=ASC(MID$(A$,I,1))
1320 IF C>64 AND C<91 THEN
C=C+32
1330 B$=B$+CHR$(C)
1340 NEXT I:RETURN
```

```
1300 LET b$=""
1305 FOR i=1 TO LEN a$
1310 LET c=ASC a$(i TO i)
1320 IF c>64 AND c<91 THEN LET
c=c+32
1330 LET b$=b$+CHR$ c
1340 NEXT i
1350 RETURN
```

Não apresentamos uma versão para os compatíveis com o ZX-81, pois estes não têm letras minúsculas.

Para testar a rotina acima, substitua a linha 30 de nosso programa de verificação por **GOSUB 1300**.

O funcionamento da rotina é simples: o laço que vai de 1305 a 1340 percorre a cadeia de entrada, tomando um caractere de cada vez, na linha 1310. Se o seu código estiver entre 65 e 90, trata-se de uma letra maiúscula; portanto, devemos somar 32 ao mesmo. A linha 130 concatena o novo caractere à cadeia de saída.

Para converter minúsculas em maiúsculas basta alterar apenas uma linha da rotina:

```
1320 IF C>96 AND C<123 THEN
C=C-32
```

A rotina verifica se o código ASCII do caractere está compreendido entre 97 e 122 (letras a a z, minúsculas). Se estiver, simplesmente subtrai 32 desse código.



Como uma cadeia é armazenada na memória do computador?

A cadeia é uma estrutura de dados muito útil, pois não exige um espaço fixo para armazenagem de cada variável literal, como acontece com as variáveis numéricas. Uma variável de precisão simples sempre ocupa quatro bytes na memória; uma variável de precisão dupla (quinze decimais) ocupa seis bytes e assim por diante.

Manter esse esquema de alocação fixa de espaço na memória para as cadeias levaria a um grande desperdício, pois o comprimento das cadeias usadas em um programa pode variar muito. Assim, o interpretador BASIC coloca todas as cadeias em um único espaço, reservado para esse fim.

Para saber onde começa e acaba cada cadeia, o interpretador mantém duas outras estruturas de dados: uma série de apontadores indica as locações de memória onde começam as cadeias. No byte 0 da cadeia está indicado o seu comprimento, em bytes. É por isso que o comprimento máximo de uma cadeia, no dialeto Microsoft BASIC, é 255 (o maior número decimal representável em oito bits).

TRS-COLOR: UM EDITOR DE DISCOS

Com o programa apresentado neste artigo, os usuários do TRS-Color terão acesso direto à informação contida nos disquetes, podendo alterá-la ou recuperá-la em caso de acidente.



A tarefa de gravar e ler arquivos em um disquete é gerenciada pelo sistema operacional de disco (DOS). Em geral, o modo como se organiza a informação no disco ou como se dá a transferência de dados não constitui problema para o usuário — o DOS cuida de tudo, transformando a unidade de disquete em uma extensão do computador.

No entanto, podemos acessar partes individualizadas do disco, manipulando e alterando diretamente a informação nele contida. Este é um recurso oferecido pelo DOS que abre inúmeras possibilidades de aplicação. Permite, por exemplo, que se alterem itens de um diretório e nomes ou dados de arquivos. Sua aplicação mais interessante talvez seja a recuperação de informações: com o acesso direto você pode, entre outras coisas, recuperar arquivos que foram apagados, fechar arquivos ou restabelecer apontadores, corrigindo o encadeamento dos setores que compõem um arquivo em particular.

O acesso direto a blocos individuais de informação em um disco funciona de modo semelhante a um monitor destinado a examinar a memória do micro.

A operação do monitor de disco (ou editor de disco) é viabilizada, basicamente, pela armazenagem temporária, em uma parte da memória, da informação lida do disco ou a ser gravada. Enquanto está na memória, a informação pode ser alterada à vontade, sendo, em seguida, substituída no disco.

A GEOGRAFIA DO DISCO

Algum conhecimento sobre a distribuição dos dados no disco é fundamental para a manipulação do bloco de informações em uma trilha ou setor determinado. O diagrama, normalmente referido como formato do disco, é definido pelas rotinas de formatação do DOS.

Usa-se a notação em hexadecimal em todas as referências sobre acesso direto ao disco — por isso, nossas referências também adotarão essa notação. Como não empregaremos a notação decimal em nosso programa, será útil ter uma tabela de conversão de decimais para hex e outra de hex para ASCII.

TRILHAS E SETORES

Para utilizar nosso programa-editor, você deverá ter um mapa do formato do disco. Veremos aqui seus aspectos mais relevantes; outros detalhes serão encontrados no manual do DOS.

Se você vai trabalhar com um disco que reúne dados importantes, recomendamos que faça uma cópia dele. Qualquer erro cometido ao usar este programa poderá danificar toda a informação nele armanezada.

O TRS-Color tem uma interface especial que contém o sistema operacional. Este não precisa, assim, estar gravado no disco para ser lido quando se liga o computador — como ocorre com a maioria dos micros.

O disco divide-se em 34 trilhas, cada uma com dezoito setores de 338 bytes. Destes, apenas 256 são usados para armazenamento de dados; os 82 restantes são utilizados para controle do sistema operacional.

O diretório, que controla os arquivos do disco, fica armazenado na trilha 17. Ele pode ser acessado por meio do comando **DIR**, que mostra todos os arquivos, discriminando seu tipo e também o número de blocos empregados. Vejamos como está dividida a trilha 17, a partir do setor 3:

Byte	Conteúdo
0-7	Nome do arquivo. Ocupando um número de bytes menor que oito, é preenchido por espaços em branco. Byte 0 igual a 0 indica arquivo apagado. Byte 0 igual a FF significa que este e os próximos itens do diretório ainda não foram usados.
8-10	Extensão do nome do arquivo.
11	Tipo de arquivo: 0 programa BASIC; 1 arquivo de dados BASIC; 2 programa em linguagem de máquina; 3 arquivo de texto.
12	00 arquivo em formato binário. FF arquivo em formato ASCII.

13 Número do primeiro bloco do arquivo (0-67).

14-15 Número de bytes em uso no último setor do arquivo.

16-31 Sem uso.

Para distribuir os arquivos no disco, o computador usa como referência *blocos*, que nada mais são que metade de uma trilha. Assim, o bloco 0 será composto pelos setores 1 a 9 da trilha 0; o bloco 1, pelos setores 10 a 18 da trilha 0; o bloco 2, pelos setores 1 a 9 da trilha 1 e assim por diante, até a trilha 16. A trilha 17 — que contém o diretório — não é levada em consideração, mas a divisão recomeça a partir da trilha 18 — onde os setores 1 a 9 compõem o bloco 34 — e, com isso, prossegue até o fim do disco...

A identificação dos blocos livres para uso é feita por meio do setor 2 da trilha 17. Os primeiros 68 bytes desse setor representam os blocos de 0 a 67. Quando determinado byte contém o valor FF, o bloco está livre. Um valor de 00 a 43 indica sua ocupação por um arquivo cuja referência é dada pelo número decimal obtido após a conversão do valor. Um valor de C0 a C9 indica que o bloco em questão é o único ou o último do arquivo; o segundo dígito especifica o número de setores que estão ocupados no bloco.

Essas informações são importantes, especialmente quando se trata de recuperar arquivos apagados por engano — tarefa, aliás, nada fácil, mas que pode ter bons resultados se realizada com um pouco de paciência.

No caso da recuperação de um arquivo pequeho, que não ocupa mais de um bloco, o trabalho é mais simples. Em primeiro lugar, você deve localizar o programa desejado no diretório. Para restaurar seu nome inicial, substitua o 00 da primeira posição pela letra adequada. Depois procure o 13º byte do item (o terceiro após a última letra da extensão do nome do arquivo). Transforme seu valor hexadecimal em decimal, para identificar o bloco usado pelo seu programa. Em seguida, grave essa informação no setor e carregue o setor 2 da trilha 17. Procure pelo byte correspondente ao bloco (lembre-se de que a contagem começa no 0). Ele deve ter o valor FF. Substitua esse valor por C9 e grave no setor. Saia do programa-editor, carregue o seu programa na memória e volte a gravá-lo, para que todos os apontadores sejam atualizados. Pronto, seu programa está totalmente recuperado!

Quando o programa que você quer recuperar é maior, a operação torna-se complicada, pois o DOS passa para FF o valor de todos os blocos usados pelo programa que foi apagado. Assim, começamos o trabalho conhecendo apenas o primeiro bloco. Os demais têm que ser procurados no disco, setor por setor. O programa só poderá ser carregado na memória do computador quando tivermos montado a sequência correta de blocos. Inicie a busca pelos blocos vizinhos do inicial e vá se afastando até ter localizado todos eles. É um trabalho árduo, mas, certamente, haverá casos em que valerá a pena.



```
10 CLEAR 5000:DIM AS(1),DS(1),D
(160):CS="^"+CHRS(10)+CHRS(8)+C
HRS(9)+"AH"+CHRS(13)+" ":D=1
15 TPS(0)="BAS" TPS(1)="DAD" TP
S(2)="LMA":TPS(3)="TXT"
20 CLS PRINT @13,"menu"
30 PRINT @105,"cARREGAR SETOR"
PRINT @169,"vER/EDITAR SETOR"·P
RINT @233,"sALVAR SETOR" PRINT
@297,"dIRETORIO"
40 R$=INKEY$:IF R$="" THEN 40
50 R=INSTR("CVSD",RS) IF R=0 TH
EN 40
60 IF SL=0 AND (R=2 OR R=3)THEN
 PRINT:PRINT"NENHUM SETOR CARRE
GADO":FOR K=1 TO 2000 NEXT GOTO
 20
70 CLS:ON R GOSUB 1000,2000,300
0,4000
80 GOTO 20
218 V$=CHR$(VAL("&H"+H$)):P=H+Y
*11+X
1000 SL=1:GOSUB 5000
1010 DSKI$ D,T,S,A$(0),A$(1)
1020 RETURN
2000 F=1·H=1·CLS:PRINT"aSCII OU
 hEX ?"
2010 R$=INKEY$:IF R$<>"A" AND R
$<>"H" THEN 2010
2020 AZ=0:IF R$="A" THEN AZ=1
2030 IF F=0 THEN 2050
2040 PK=96 CP=1535:IF AZ=1 GOSU
B 2320 ELSE GOSUB 2280
2050 POKE CP,PK:CP=1024+Y*32+X*
3:PK=PEEK(CP) POKE CP,239
2060 PRINT @321,"BYTE SUPERIOR=
";H
2070 R$=INKEY$ IF R$="" THEN 20
70
2080 R=INSTR(C$,R$)·IF R=0 THEN
 2070
2090 F=0:ON R GOTO 2100,2110,21
20,2130,2140,2150,2160,2170
2100 Y=Y 1 GOTO 2210
2110 Y=Y+1·GOTO 2210
2120 X=X-1.GOTO 2210
2130 X=X+1·GOTO 2210
2140 AZ=1·GOTO 2040
2150 AZ=0:GOTO 2040
2160 RETURN
2170 PRINT @384,"NOVO CONTEUDO
(HEX) ";:INPUT H$
2180 V$=CHR$(VAL("&H"+H$)):P=H+
Y*11+X
2190 MID$(A$(P/128),P+128*(P>12
8),1)=V$
2200 F=1:GOTO 2030
2210 IF Y<0 THEN H=H-44:Y=0:F=1
2220 IF Y>7 THEN H=H+44 Y=7·F=1
2230 IF X<0 THEN X=10:Y=Y-1:IF
Y<0 THEN H=H-11:Y=0:F=1
2240 IF X>10 THEN X=0.Y=Y+1.IF
Y>7 THEN Y=7:H=H+11.F=1
2250 IF H= 10 OR H=-43 THEN H=1
:F=0·ELSE IF H<1 THEN H=1 F=1
2260 IF H=179 OR H=212 THEN H=1
68:F=0 ELSE IF H>168 THEN H=168
:F=1
2270 GOTO 2030
2280 CLS:FOR J=H TO H+87 STEP 1
1 FOR TT=0 TO 10
2290 PRINT RIGHT$("0"+HEX$(ASC(
MID$(A$(J/128),J+TT+128*((J+TT)
>128)))),2);" ";
2300 NEXT:PRINT CHR$(8);·NEXT
2310 RETURN
2320 CLS:FOR J=H TO H+87 STEP 1
1:FOR TT=0 TO 10
2330 G=ASC(MID$(A$(J/128),J+TT+
128*((J+TT)>128)))·IF G<32 THEN
 2350
2340 PRINT " ",CHR$(G);" ",:GOT
O 2360
2350 PRINT LEFT$("0"+HEX$(G),2)
;" ";
2360 NEXT.PRINT CHR$(8); NEXT:R
ETURN
3000 CLS:PRINT"SALVAR NO MESMO
SETOR (S/N) ?"
3010 R$=INKEY$:IF R$<>"S" AND R
$<>"N" THEN 3010
3020 IF R$="S" THEN 3040
030 CLS·GOSUB 5000
```

```
3040 PRINT:PRINT"VOCE TEM CERTE
ZA (S/N) ?"
3050 R$=INKEY$ IF R$<>"S" AND R
$<>"N" THEN 3050
3060 IF R$="N" THEN RETURN
3070 DSKO$ D,T,S,A$(0),A$(1)
3080 RETURN
4000 GOSUB 5050
4010 PRINT #PR,TAB(14);"INICIO
NO."
4020 PRINT #PR,"  NOME     EXT T
IPO    DEL"
4030 FOR J=0 TO 8:DSKI$ D,17,J+
3,D$(0),D$(1)
4040 FOR K=1 TO 256 STEP 32
4050 GOSUB 6000
4060 IF ASC(V$)=255 THEN K=256:
J=8:GOTO 4120
4065 IF ASC(V$)=0 THEN MID$(V$,
1,1)=" " DT=1
4070 PRINT #PR,MID$(V$,1,8);TAB
(9);".";MID$(V$,9,3);
4080 TP=VAL(MID$(V$,11,1))
4100 PRINT #PR,TAB(15);TP$(TP);
TAB(20)CHR$(42*DT)
4110 DT=0
4120 NEXT K,J:R$=INKEY$:IF PR=-
2 THEN 4140
4130 R$=INKEY$:IF R$="" THEN 41
30
4140 RETURN
5000 INPUT"NUMERO DA TRILHA (0-
34) ";T
5010 INPUT"NUMERO DO SETOR (1-1
8) ";S
5020 INPUT"NUMERO DO DRIVE (0-3
) ";D
5030 IF D>3 OR D<0 OR T>34 OR T
<0 OR S>18 OR S<1 THEN 5000
5040 RETURN
5050 PR=0:IF(PEEK(150))<>1 THEN
 RETURN
5060 PRINT"SAIDA PARA A IMPRESS
ORA (S/N)?"
5070 R$=INKEY$:IF R$<>"S" AND R
$<>"N" THEN 5070
5080 IF R$="S" THEN PR=-2
5090 RETURN
6000 V$=MID$(D$(K/128),K+128*(K
>128),32):IF LEN(V$)<32 THEN V$
=V$+MID$(D$(1+K/128),1,32-LEN(V
$))
6010 RETURN
```



COMO USAR O PROGRAMA

Carregue o programa na memória e escolha o disco no qual você vai trabalhar. Enquanto está aprendendo, convém utilizar um disquete que não contenha dados importantes.

Digitando **RUN**, você terá quatro opções: *Carregar setor* (**C**), *Ver/editar setor* (**V**), *Salvar setor* (**S**), *Diretório* (**D**). Pressione **D** para obter uma listagem detalhada dos arquivos do disco, incluindo os que foram apagados anteriormente. Estes estarão marcados com um asterisco na coluna DEL. Lembre-se de que a inclusão do nome de um arquivo no diretório não garante que ele esteja disponível no disco — os blocos usados por ele podem ter sido reutilizados por um outro arquivo. Se você tem uma impressora conectada ao Micro, digite **POKE 150,1** antes de executar o programa: a opção de imprimir o diretório lhe será oferecida.

Em seguida, digite a tecla **C** para carregar um setor. Tente primeiro o diretório, pedindo a trilha 17, setor 3. Digite **V** para visualizar os dados. Pressionando A ou H, você poderá obter, a qualquer momento, os dados no formato ASCII ou hexadecimal. O primeiro é o mais conveniente para a identificação de nomes de arquivos e linhas BASIC. O segundo deve ser usado quando se quer encontrar os valores hexadecimais de um determinado byte.

As setas movem o cursor para o ponto que se desejar. Para alterar um valor, pressione a barra de espaços e forneça o novo byte, sempre em hexadecimal. Note que apenas uma parte do setor é exibida. À medida que se movimenta o cursor para baixo, a tela vai rolando e mostrando o que resta.

Ao terminar a edição de um setor, pode-se gravá-lo usando a opção **S** do menu. Entre as informações solicitadas pelo programa... e pronto!

TÉCNICAS DE RECURSÃO

- UMA TÉCNICA MISTERIOSA?
- A MÁQUINA INTELIGENTE
- PROCEDIMENTOS RECURSIVOS
- LIMITAÇÕES
- COMO EVITAR ERROS

Se seu programa possui sub-rotinas que são chamadas repetidamente, é bem provável que as técnicas de recursão possam torná-lo mais rápido e eficiente. Veja como utilizá-las.

A construção de um programa consiste, essencialmente, em achar a melhor solução para um determinado problema através de uma série de comandos. À medida que ganha experiência, o programador aprende que o melhor caminho para isso é dividir o problema principal em partes menores. Tal método, no entanto, pode virar um pesadelo. Muitas vezes, ao tentar resolver uma etapa do processo, esbarra em outro processo e, quando resolve este, surge ainda outro e assim por diante. Em certo momento, o programador pensa em desistir, pois imagina estar andando em um verdadeiro círculo vicioso.

Mas um microcomputador não esmoreceria tão facilmente: para casos desse tipo, em que os problemas parecem sair uns dos outros, existem técnicas avançadas de programação, as chamadas *técnicas de recursão*.

UMA TÉCNICA MISTERIOSA?

Geralmente, programadores iniciantes consideram a recursão uma ferramenta complexa e até mesmo misteriosa. A grande dificuldade está na compreensão da própria listagem de um programa desse tipo; até mesmo os fluxogramas mais didáticos parecem obscuros quando representam processos recursivos. Apesar de tudo, o princípio básico dessa técnica não é tão complicado.

Na matemática, a palavra recursão é utilizada para designar a repetição de uma determinada operação. Em informática, porém, ela adquire um significado específico. Basicamente, a recursão é o chamamento sucessivo de uma sub-rotina ou de um procedimento, após serem estabelecidos alguns parâmetros iniciais. Depois de acionada, a sub-rotina — ou procedimento — chama a si mesma, repetidamente, atualizando os



parâmetros iniciais a cada vez, até que uma certa condição, previamente estabelecida, seja alcancada.

A MÁQUINA INTELIGENTE

Imaginemos a seguinte situação: um motorista se encontra em uma grande cidade desconhecida e precisa ir do lugar A para o lugar B. Embora tenha o mapa da cidade, acha muito difícil ir diretamente ao local desejado. Decide então dividir o itinerário em vários trechos, estudando um por vez. Escolhe, assim, um lugar C, em uma posição intermediária. Analisa o trajeto entre A e C e se dirige para lá.

Chegando em C, o motorista avalia a possibilidade de ir diretamente até B Caso isso lhe pareça complicado, procura uma outra posição intermediária, D. O processo se repete até que ele atinja seu destino.

Esse exemplo demonstra o princípio básico da recursão, tal como é aplicada em programação: a solução de um problema é obtida pela sua subdivisão em problemas menores ou mais fáceis.

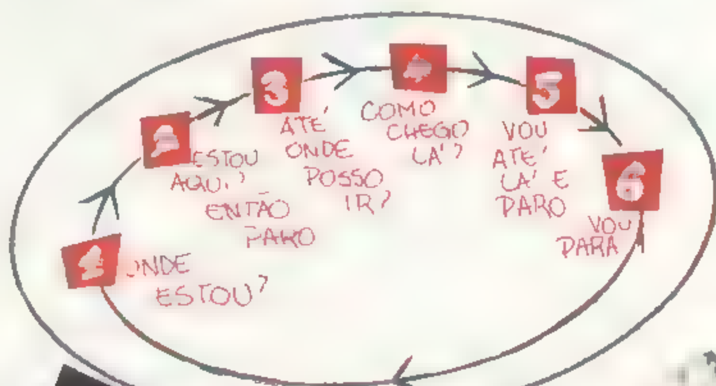
Assim que o programa alcança um novo nível de recursão, torna-se necessário armazenar as informações que foram conseguidas no nível anterior, para recuperá-las no passo seguinte. Ao início de cada nível, um novo conjunto de parâmetros é montado e, por meio de um teste, verifica-se se tal etapa chegou ao fim. Sem esse teste, o processo nunca terminaria.

Para observar o funcionamento dessa técnica, execute o próximo programa. Ele imprime todos os valores inteiros positivos de N até 1.

```
 20 PRINT  INVERSE 1;TAB 1;" I
NTEIROS POSITIVOS DE N ATE 1"
 30 INPUT "DIGITE O INTEIRO A
PARTIR DO    QUAL VOCE QUER C
ONTAR REGREDIN- DO ATE 1 (0 P
ARA SAIR) ";N: LET N=INT N:
IF N<1 THEN  STOP
 40 GOSUB 80
 50 GOTO 30
 80 IF N=0 THEN  RETURN
 90 PRINT ;N;"  ";
100 LET N=N-1: GOSUB 80
110 RETURN
```

```
10 CLS
20 PRINT "INTEIROS POSITIVOS DE
 N A 1"
30 PRINT:PRINT INPUT"DIGITE O V
ALOR INTEIRO A PARTIR DO QUAL V
OCE QUER REGREDIR ATE 1(0 OU  N
EGATIVO PARA TERMINAR) ";N:N=IN
T(N):IF N<1 THEN END
40 GOSUB 80
50 GOTO 30
80 IF N=0 THEN RETURN
90 PRINT N;".";
100 N=N-1:GOSUB 80
110 RETURN
```

```
10  HOME
20  PRINT  TAB( 7)"INTEIROS POS
```



```
ITIVOS DE 1 A N"
30  PRINT : PRINT · INPUT "DIGI
TE UM VALOR INTEIRO PARA A CONT
AGEM REGRESSIVA (1-23 E 0 OU ME
NOS PARA SAIR) ";N:N =  INT (N)
: IF N < 1 THEN  END
40  GOSUB 80
50  GOTO 30
80  IF N = 0 THEN  RETURN
90  PRINT N;",";
100 N = N - 1· GOSUB 80
110  RETURN
```



```
10 CLS
20 PRINT TAB(5) "INTEIROS POSIT
IVOS DE 1 A N"
30 PRINT·PRINT:INPUT"DIGITE O N
UMERO PARA A CONTAGEM REGRES-SI
VA (0 OU MENOS PARA SAIR)";N:N=
INT(N)·IF N<1 THEN END
40 GOSUB 80
50 GOTO 30
80 IF N=0 THEN RETURN
90 PRINT N;",";
100 N=N-1:GOSUB 80
110 RETURN
```

O programa permite que você introduza um valor inteiro e, a partir dele, faz uma contagem regressiva até 1. Qualquer valor menor que 1 interrompe o programa. No Apple e no TK-2000, valores maiores que 23 não funcionarão, pois esses micros podem memorizar apenas 23 desvios para uma sub-rotina

A linha 40 chama a sub-rotina recursiva. A primeira linha desta verifica se a tarefa foi completamente executada. Esse teste é essencial para o término de todo o processo.

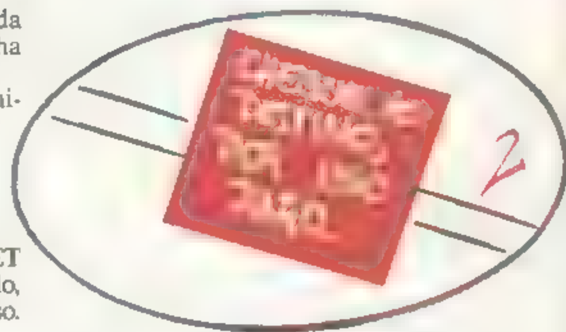
No primeiro nível da recursão, a variável **N** possui o valor introduzido por você. Ele é impresso pela linha 90. O segundo nível é inicializado pela linha 100, que reduz o valor de **N** em uma unidade e chama a sub-rotina novamente, com o valor de **N** já alterado. O programa alterna-se, assim, repetidamente, entre as linhas 80 e 100.

Quando **N** alcança o valor 0 (na linha 100), o programa é desviado, como de costume, para a linha 80, onde encontra o comando **RETURN**, que o faz retornar da sub-rotina chamada na linha 100. A próxima instrução, na linha 110, também é um comando **RETURN**, que, desta vez, faz o programa retornar da sub-rotina chamada na linha 40. A linha 50 executa o programa novamente.

Note que quando o programa termina, a variável **N** contém o valor 0, atribuído na linha 100. Mas a linha 90 nunca imprime esse valor. Para colocar o último valor impresso na variável **N**, você poderia acrescentar a linha 105 com o comando **N=N+1** (**LET N=N+1**, para o Spectrum). Desse modo, **N** conteria sempre o último valor impresso.

PROCEDIMENTOS RECURSIVOS

Algumas outras linguagens, como o PASCAL, incluem o que se chama de *procedimento* (*procedure*). Trata-se de um conjunto de linhas definidas em uma certa parte do programa — como uma sub-rotina — que recebe um nome. Diferentemente das sub-rotinas, um procedimento é chamado através desse nome. Quando se define um procedimento, indicam-se também as variáveis que, após seu término, retornarão ao estado inicial. Este é um bom expediente na construção de processos recursivos.



Os tipos de BASIC com que trabalhamos não permitem que uma variável assuma um valor e, ao sair da sub-rotina, retorne ao valor inicial sem a interferência de um comando. Assim, somos obrigados a restabelecer seus valores iniciais antes de iniciar outra sub-rotina. O programa a seguir demonstra como contornar esse problema.



```
10 DIM N(34): DIM A(34)
20 CLS
```

```
30 PRINT TAB 6; INVERSE 1;"C
ALCULO DE FATORIAL"
40 INPUT "DIGITE NUMERO PARA
FATORIAL      (1-33, OU 0 PARA
 SAIR)";NU
50 IF NU>33 OR NU<>INT (NU)
OR NU<0 THEN  RUN
60 IF NU=0 THEN  STOP
70 LET LE=1: LET N(LE)=NU:
LET AN=NU
80 GOSUB 150
90 PRINT AN;"! = ";A(1):
PRINT : GOTO 40
150 IF N(LE)=0 THEN  LET A(LE)
=1: GOTO 180
160 LET LE=LE+1: LET N(LE)=N(L
E-1)-1: GOSUB 150
170 LET LE=LE-1: LET A(LE)=A(L
E+1)*N(LE)
180 RETURN
```



```
10 DIM N(34),A(34)
20 CLS
30 PRINT @6,"CALCULO DE FATORIA
L"
40 INPUT"DIGITE O NUMERO PARA F
ATORIAL   (1-33 OU 0 PARA SAIR)
   ";NU
50 IF NU>33 OR NU<>INT(NU) OR N
U<0 THEN 20
60 IF NU=0 THEN END
70 LE=1:N(LE)=NU:AN=NU
80 GOSUB 150
90 PRINT AN"! = ";A(1):PRINT:GO
TO 40
150 IF N(LE)=0 THEN A(LE)=1:GOT
O 180
160 LE=LE+1:N(LE)=N(LE-1)-1:GOS
UB 150
170 LE=LE-1:A(LE)=A(LE+1)*N(LE)
180 RETURN
```



```
10  DIM N(34),A(34)
20  HOME
30  PRINT  TAB( 10)"CALCULO DE
FATORIAIS"
40  PRINT : INPUT "DIGITE O NUM
ERO FATORIAL (1-22 OU 0 PARASAI
R) ";NU
50  IF NU > 22 OR NU <  >  INT
(NU) OR NU < 0 THEN 20
60  IF NU = 0 THEN  END
70 LE = 1:N(LE) = NU:AN = NU
80  GOSUB 150
90  PRINT AN"! = ";A(1): PRINT
: GOTO 40
150  IF N(LE) = 0 THEN A(LE) =
1: GOTO 180
160 LE = LE + 1:N(LE) = N(LE -
1) - 1: GOSUB 150
170 LE = LE - 1:A(LE) = A(LE +
1) * N(LE)
180  RETURN
```



```
10 DIM N(34),A(34)
20 CLS
30 PRINT TAB(9) "CALCULO DE FAT
ORIAIS"
40 PRINT:INPUT" DIGITE O NUMERO
 FATORIAL (1-33 E 0 OU MENOS PA
RA SAIR) ";NU
50 IF NU>33 OR NU<>INT(NU) OR N
U<0 THEN 20
60 IF NU=0 THEN END
70 LE=1:N(LE)=NU:AN=NU
80 GOSUB 150
90 PRINT:PRINT AN;"! = ";A(1):P
RINT:GOTO 40
150 IF N(LE)=0 THEN A(LE)=1:GOT
O 180
160 LE=LE+1:N(LE)=N(LE-1)-1:GOS
UB 150
170 LE=LE-1:A(LE)=A(LE+1)*N(LE)
180 RETURN
```



Execute o programa e introduza um valor qualquer. Lembre-se de que valores maiores que 22 não funcionarão no Apple, nem no TK-2000.

O programa calcula o fatorial do número que você digitou e o imprime na tela. O fatorial de um número é o resultado da multiplicação, entre si, de todos os números inteiros menores que ele e ele mesmo. Por exemplo, 5 fatorial (escreve-se 5!) é igual a 1 x 2 x 3 x 4 x 5, ou seja, 120. O cálculo de fatoriais geralmente é necessário em aplicações estatísticas. Um método rápido de executá-lo será sempre muito útil.

Inicialmente, o programa dimensiona algumas variáveis (linha 10), em número suficiente para gerar o fatorial máximo. A variável **N** terá como índice o nível da recursão (**LE**) — **N(LE)**.

A estrutura principal do programa começa na linha 70, onde o nível é inicializado em 1. Nessa mesma linha, o número que você introduziu — suponhamos que seja 5 — é atribuído a **N(1)** e a **AN** (variável que irá acumular a resposta). A linha 80 chama então o primeiro nível da recursão. A primeira linha da rotina de recursão (linha 150) verifica se a tarefa já chegou ao fim, ou seja, se **N(LE)** = 0. Como **N(LE)** ainda é 5, o programa passa para a linha 160, que incrementa o nível (para 2), estabelece como número corrente o 4 e chama a sub-rotina recursiva mais uma vez. Quando a linha 160 incrementar o valor do nível para 6 e decrementar o número corrente para 0, a linha 150 detectará esse último valor atribuindo 1 a **N(6)** e desviando o programa para a linha 180. Encontrando um **RETURN**, a execução passará para a linha 170, que foi a última a chamar a sub-rotina. Essa linha reduzirá o valor do nível para 5 e **A(5)** passará a ter o valor de **A(6)** vezes **A(5)**. Com isso, atribui-se a **A(5)** o valor de 1 x 1, ou seja, 1. A linha 180 devolve, então, o controle para o final da linha 150, onde **A(4)** assume o valor de **A(5)** vezes **A(4)**. Desse modo, será atribuído ao nível da recursão o valor 1 (calculado acima) vezes 2.

O laço irá se repetir sempre que o **GOSUB** da linha 160 for chamado. Quando o laço estiver completo, **LE** será 1 e o último **RETURN** apontará para a linha 80. A próxima instrução (linha 90) imprimirá então o resultado: 120

LIMITAÇÕES

Além do problema com as variáveis, que não ocorre com os procedimentos, os tipos de BASIC com os quais trabalhamos apresentam uma limitação quanto ao número de vezes que uma sub-rotina pode chamar outra. A cada chamada o sistema precisa guardar na memória a posição de onde ela foi feita, o que requer a utilização de um ponteiro na pilha interna. É por esse motivo que o Apple e o TK-2000 não podem calcular fatoriais maiores que 22. Já o limite de 33 níveis para outros micros deve-se simplesmente à capacidade de armazenamento de números. O fatorial de 34 excede 1.7 x 10 ↑ 38, o máximo que um computador pode manipular.

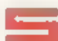
PREVISÃO DE ERROS

Para manter seu programa dentro dos limites do micro e evitar uma "pane", você deve saber como ele se comporta desde o primeiro nível da recursão. Em geral, utiliza-se como teste uma informação inicial que culminará em uma resposta já esperada. Para facilitar o trabalho, você pode também considerar a rotina recursiva como um certo número de sub-rotinas em seqüência. Note que um desvio condicional para fora da sub-rotina é desaconselhável, pois o ponteiro que marca a volta do **GOSUB** ficaria desorientado.

Quando você estiver escrevendo uma sub-rotina recursiva, procure começar sempre com um teste de saída. A função desse teste é decidir se o problema (caracterizado pelos parâmetros iniciais) pode ser resolvido diretamente, sem a necessidade de subdivisões.

Antes de começar o programa, planeje com cuidado o que deseja que a sub-rotina faça. E, quando já estiver escrevendo o programa, não se preocupe com a seqüência exata da execução, levando em conta apenas dois princípios básicos: a condição de saída e a subdivisão em problemas menores.

Seguindo o método aqui apresentado, você será capaz de aplicar a recursão aos seus programas sem maiores dificuldades. Aqui está um exemplo de como essa técnica pode melhorá-los, tornando-os assim mais rápidos e compactos



```
10 BORDER 1: INK 7: PAPER 1:
CLS
20 PRINT TAB 11; INVERSE 1;"
ORDENACAO "
30 INPUT "QUANTOS NUMEROS VOC
E QUER        ORDENAR (1-1000)
";A
40 IF A<1 OR A>1000 THEN
GOTO 10
50 DIM A(A): DIM R(2+SQR (A))
60 LET A(A)=100: PRINT
INVERSE 1;'"TABELA DESORDENAD
A :"'': FOR K=1 TO A-1: LET A
(K)=INT (RND*99): PRINT A(K);
" ";: NEXT K
70 LET L=1: LET LV=1: LET R=A
-1: GOSUB 1000
80 PRINT  INVERSE 1;''"TABELA
 ORDENADA :"'': FOR K=1 TO A-
1: PRINT A(K);" ";: NEXT K
90 IF INKEY$<>" " THEN  GOTO
 90
100 RUN
1000 IF R>L THEN  LET I=L· LET
J=R+1: LET V=A(L): GOTO 1010
1005 RETURN
1010 LET I=I+1: IF A(I)<V THEN
 GOTO 1010
1020 LET J=J-1: IF A(J)>V THEN
 GOTO 1020
1030 IF J>=I THEN  LET T=A(I):
LET A(I)=A(J)· LET A(J)=T: GOTO
 1010
1040 LET T=A(L): LET A(L)=A(J):
 LET A(J)=T
1050 LET R(LV)=R: LET LV=LV+1:
LET R=J-1: GOSUB 1000
1060 LET LV=LV-1: LET R=R(LV)·
LET L=1: GOSUB 1000
1070 RETURN
```



```
10 CLS
20 PRINT @11,"ORDENACAO"
30 PRINT INPUT" QUANTOS NUMEROS
 VOCE QUER        ORDENAR  (1-10
00)  ";A
40 IF A<1 OR A>1000 THEN 10
50 DIM A(A),R(1+SQR(A))
60 A(A)=100:PRINT" TABELA DESOR
DENADA :":FOR K=0 TO A-1·A(K)=R
ND(99):PRINT A(K), NEXT:PRINT
70 L=0·R=A-1·GOSUB 1000
80 PRINT " TABELA ORDENADA ·":F
OR K=0 TO A-1:PRINT A(K);:NEXT
90 IF INKEY$<>" " THEN 90:ELSE
RUN
1000 IF R>L THEN I=L:J=R+1:V=A(
L) ELSE RETURN
1010 I=I+1·IF A(I)<V THEN 1010
1020 J=J-1·IF A(J)>V THEN 1020
1030 IF J>=I THEN T=A(I):A(I)=A
(J):A(J)-T:GOTO 1010
1040 T=A(L):A(L)=A(J):A(J)=T
1050 R(LV)=R:LV=LV+1:R=J-1:GOSU
B 1000
1060 LV=LV-1:R=R(LV)·L=I·GOSUB
1000
1070 RETURN
```



```
10  HOME
20  HTAB (11): INVERSE : PRINT
" ORDENANDO NUMEROS ": NORMAL
30  PRINT : INPUT " QUANTOS NUM
EROS A SEREM ORDENADOS ? (1-100
0) ";A
```

```
40  IF A < 1 OR A > 1000 THEN 1
0
50  DIM A(A),R(1 +  SQR (A))
60 A(A) = 100: PRINT : PRINT "
NUMEROS FORA DE ORDEM :-";: FOR
 K = 0 TO A - 1:A(K) =  INT (99
 *  RND (1)) + 1: PRINT A(K);"
";: NEXT : PRINT
70 L = 0:LV = 0:R = A - 1: GOSU
B 1000
80  PRINT " NUMEROS NA ORDEM :-
";: FOR K = 0 TO A - 1: PRINT A
(K);" ";: NEXT
90  GET I$: IF I$ < > " " THEN
 90
100  RUN
1000  IF R > L THEN I = L:J = R
 + 1·V = A(L): GOTO 1010
1005  RETURN
1010 I = I + 1: IF A(I) < V THE
N 1010
1020 J = J - 1: IF A(J) > V THE
N 1020
1030  IF J >  = I THEN T = A(I)
·A(I) = A(J):A(J) = T· GOTO 101
0
1040 T = A(L):A(L) = A(J):A(J)
= T
1050 R(LV) = R:LV = LV + 1:R =
J - 1: GOSUB 1000
1060 LV = LV - 1:R = R(LV):L =
I: GOSUB 1000
1070  RETURN
```

```
10 CLS
20 PRINT TAB(10)"ORDENAÇAO DE N
UMEROS"
30 PRINT·INPUT"QUANTOS NUMEROS
SERAO ORDENADOS(1-1000)";A
40 IF A<1 OR A>1000 THEN 10
50 DIMA(A),R(1+SQR(A))
60 A(A)=100·PRINT"NUMEROS FORA
DE ORDEM·-"·FOR K=0 TO A-1:A(K)
=INT(RND(-TIME)*99)+1·PRINT A(K
);·NEXT·PRINT
70 L=0·LV=0 R=A-1:GOSUB 1000
80 PRINT"NUMEROS ORDENADOS·"·FO
R K=0 TO A-1·PRINT A(K);:NEXT
90 IF INKEY$<>" " THEN 90 ELSE
RUN
1000 IF R>L THEN I=L:J=R+1:V=A(
L):GOTO 1010
1005 RETURN
1010 I=I+1:IF A(I)<V THEN 1010
1020 J=J-1:IF A(J)>V THEN 1020
1030 IF J>=I THEN T=A(I):A(I)=A
(J)·A(J)=T:GOTO 1010
1040 T=A(L):A(L)=A(J):A(J)=T
1050 R(LV)=R:LV=LV+1:R=J-1:GOSU
B 1000
1060 LV=LV-1:R=R(LV):L=I:GOSUB
1000
1070 RETURN
```

PROGRAMA DE ORDENAÇÃO

Compare a listagem do método de ordenação tipo bolha do artigo da página 468 com esta, que utiliza a recursão. Nosso programa não contém tantos desvios condicionais do tipo **IF...THEN, GOTO...**, nem tantas variáveis.

Inicialmente, o programa pede que você introduza a quantidade de números randômicos que serão ordenados. A linha 50 dimensiona a matriz **A(A)**, que armazena esses números, e a matriz **R**, que irá guardar o valor das variáveis durante o processo recursivo. A linha 60 gera e imprime os números randômicos, ainda fora de ordem, e a linha 70 chama a sub-rotina recursiva para ordená-los. A linha 80 é responsável pela impressão final.

O método baseia-se na junção de duas listas, ambas previamente submetidas a uma ordenação. A lista principal, composta de números randômicos desordenados, é dividida em duas outras listas (linhas 1010 e 1020). Observe que, na linha 1000, há um teste de saída para determinar o fim da recursão. Cada uma das listas sofre então uma ordenação parcial (linha 1030), sendo posteriormente reunidas. A sub-rotina irá chamar a si mesma (linha 1050) até que a ordenação esteja completa.

Apesar dos métodos de ordenação em BASIC não serem tão rápidos quanto os que empregam linguagem de máquina, este programa pode ser muito eficiente em tarefas menos extensas. Por exemplo, a ordenação de cem números, no Spectrum, leva mais ou menos 40 segundos; a ordenação tipo bolha nos faria esperar mais de uma hora.

APLICAÇÕES

A utilidade da recursão vai muito além do cálculo de fatoriais e outras funções matemáticas. Ela pode ser aplicada na programação de jogos e na construção dos mais complexos gráficos. Essa técnica também aparece em sistemas de inteligência artificial e no controle de robôs, além de ser empregada em processamento de linguagens (compiladores e interpretadores).

Os jogos de estratégia, como o xadrez ou os *wargames*, constituem uma outra área interessante de aplicação. O programa que apresentamos a seguir usa a recursão na simulação de um jogo clássico, *As Torres de Hanói*.

```
10 BORDER 6: PAPER 6: INK 0:
CLS
20 PRINT TAB 8; INVERSE 1;"
TORRES DE HANOI "
30 INPUT "DIGITE O NUMERO DE
ANEIS (2-9)□";N: IF N<2 OR N
>9 THEN  GOTO 30
35 DIM T(3)
36 LET A$="■  ■": INK 2: FOR M
=21 TO 21-N STEP -1: PRINT AT
M,7;A$;AT M,15;A$;AT M,23;A$:
NEXT M: INK 0
37 PRINT  INK 2;AT 21,7;"■■■"
;AT 21,15;"■■■";AT 21,23;"■■
■"
38 FOR M=1 TO N: PRINT  INK 7
; PAPER 0;AT 20-T(1),8;N+1-M:
LET T(1)=T(1)+1: NEXT M
39 PRINT #1;AT 0,3;"QUALQUER
TECLA PARA COMECAR"
40 PAUSE 0: PRINT #1;AT 0,3;"
□          □□          □ "
45 LET TT=2: LET TF=1: LET R=
3
50 GOSUB 90
70 PRINT AT 10,5;"NUMERO DE M
OVIMENTOS=□";2^N-1
80 STOP
90 IF N=0 THEN  RETURN
100 LET N=N-1: LET W=R: LET R=
TT: LET TT=W: GOSUB 90: LET W=
R: LET R=TT: LET TT=W
110 GOSUB 200
120 LET W=R: LET R=TF: LET TF=
W: GOSUB 90: LET W=R: LET R=TF
: LET TF=W
130 LET N=N+1: RETURN
200 PRINT AT 20-(T(TF)-1),TF*8
;" "; INVERSE 1;AT 20-(T(TT)),
TT*8;N+1: LET T(TF)=T(TF)-1:
LET T(TT)=T(TT)+1
210 SOUND .01,TT*T(TT)*2
220 RETURN
```

```
10 CLS:DIM H(3)
20 PRINT @8,"TORRES DE HANOI":P
RINT
30 PRINT"NUMERO DE ANEIS (2-9)
?";
40 A$=INKEY$:IF A$<"2" OR A$>"9
" THEN 40
50 N=VAL(A$):H(0)=N:PRINT @64
60 FOR K=0 TO 8:FOR J=0 TO 2:PR
INT @165+K*32+J*9,CHR$(175)+" "
+CHR$(175);:NEXT J,K
70 FOR K=0 TO 2:PRINT @453+K*9,
STRING$(3,175);:NEXT
80 FOR K=1 TO N:POKE 1478-32*K,
49+N-K:NEXT
90 TT=1:TF=0:R=2
100 GOSUB 1000
110 PRINT @65,"NUMERO DE MOVIME
NTOS =";INT(2^N-1)
120 PRINT" QUALQUER TECLA PARA
RECOMECAR"
130 IF INKEY$="" THEN 130
140 RUN
1000 IF N=0 THEN RETURN
1010 N=N-1:W=R:R=TT:TT=W:GOSUB
1000:W=R:R=TT:TT=W
1020 POKE 1478+9*TF-32*H(TF),96
:H(TF)=H(TF)-1:H(TT)=H(TT)+1:PO
KE 1478+9*TT-32*H(TT),49+N
1030 W=R:R=TF:TF=W:GOSUB 1000:W
=R:R=TF:TF=W
1040 N=N+1:RETURN
```



```
10  HOME : DIM T(3)
20  HTAB (14): INVERSE : PRINT "TORRE DE HANOI": NORMAL
30  PRINT : PRINT : INPUT " NUMERO DE ANEIS (2-9): ";N
40  IF N < 2 OR N > 9 THEN  RUN
50  FOR J = 1 TO 3: INVERSE : VTAB (16): HTAB (J * 10): PRINT J: NEXT : NORMAL
60  FOR I = 1 TO N: HTAB (10): VTAB (16 - I): PRINT N - I: NEXT I
70 T(1) = 15 - N:T(2) = 15:T(3) = 15
80 TT = 2:TF = 1:R = 3: GOSUB 110
90  HTAB (14): VTAB (20): PRINT "MOVIMENTOS=";2 ^ N - 1
100  END
110  IF N = 0 THEN  RETURN
120 N = N - 1:W = R:R = TT:TT = W: GOSUB 110:W = R:R = TT:TT = W
130  GOSUB 160
140 W = R:R = TF:TF = W: GOSUB 110:W = R:R = TF:TF = W
150 N = N + 1: RETURN
160 T(TF) = T(TF) + 1: HTAB (TF * 10): VTAB (T(TF)): PRINT " "
170  HTAB (TT * 10): VTAB (T(TT)): PRINT N:T(TT) = T(TT) - 1
180  RETURN
```



```
10 KEY OFF:CLS:DIM T(3)
20 PRINT TAB(13)"TORRE DE HANÓI"
30 PRINT:PRINT:INPUT" NUMERO DE ANEIS (2-9): ";N
40 IF N<2 OR N>9 THEN RUN
50 FOR I=1 TO N:LOCATE 10,16-I:PRINT N-I:NEXT I
60 T(1)=15-N:T(2)=15:T(3)=15
70 TT=2:TF=1:R=3:GOSUB 100
80 LOCATE 14,20:PRINT" MOVIMENTOS=";2^N-1
90 END
100 IF N=0 THEN RETURN
110 N=N-1:W=R:R=TT:TT=W:GOSUB 100:W=R:R=TT:TT=W
120 GOSUB 150
130 W=R:R=TF:TF=W:GOSUB 100:W=R:R=TF:TF=W
140 N=N+1:RETURN
150 T(TF)=T(TF)+1:LOCATE TF*10,T(TF):PRINT"  "
160 LOCATE TT*10,T(TT):PRINT N:T(TT)=T(TT)-1
170 RETURN
```

Quando o jogo começa, vários discos de diâmetros diferentes acham-se empilhados sobre a primeira de três varetas montadas sobre o tabuleiro. O objetivo é passar todos os discos para a segunda vareta, movendo um por vez e sem deixar um disco maior sobre um menor. A terceira vareta serve como apoio temporário durante o processo.

O computador mostrará uma simulação animada, pedindo, inicialmente, o número de discos que sairão da primeira vareta. A transferência destes será bastante rápida, o que dificulta a visualização dos movimentos. Caso queira observar melhor o processo, faça as modificações que se seguem. Para o Spectrum, introduza esta linha:

```
215 PAUSE 0
```

Para o TRS-Color, adicione este comando ao final da linha 1020:

```
:SOUND 50+H(TT)*10,12
```

No Apple e no TK-2000, acrescente:

```
175 FOR I=1 TO 800:NEXT
```

Para o MSX, adicione esta linha:

```
165 BEEP:FOR I=1 TO 800:NEXT
```

O programa é semelhante aos anteriores. A rotina recursiva executa cada movimento sucessivamente, até que todos os discos estejam na segunda vareta. No fim, o computador também exibe o total de movimentos realizados.



As Torres de Hanói **e seus nove discos.**

Apesar de poderosa, a recursão pode ser inconveniente quando há uma grande limitação de memória, roubando mais espaço e mais tempo do que o necessário. Porém, se não há tal impedimento e se em seu programa uma subrotina é chamada mais de duas vezes, procure aperfeiçoá-lo com essa técnica.

AVALANCHE: PONTOS GANHOS

Nem tudo é desastre no caminho de Willie. Para que não se desespere totalmente, às vezes é recompensado, conseguindo chegar ao topo da montanha e recuperar seu lanche.

Nenhum outro aventureiro conheceu tantos infortúnios quanto Willie. Nosso personagem já experimentou até a morte e a descida para o inferno. Merece, agora, uma recompensa: finalmente, ele conseguirá alcançar o topo da montanha e recuperar mais uma parte do lanche perdido, conquistando, assim, alguns pontos no placar.

S

A pequena rotina inicial executa a melodia da recompensa, coloca Willie no próximo nível, aumenta a velocidade do jogo e incrementa o placar.

```
 10 REM org 59788
 20 REM rwd ld de,523
 30 REM ld hl,806
 40 REM call 949
 50 REM ld a,(57344)
 60 REM inc a
 70 REM res 2,a
 80 REM ld (57344),a
 90 REM ld a,(58732)
100 REM dec a
110 REM ld (58732),a
120 REM ld a,2
130 REM ld b,5
140 REM call 59900
150 REM jp 58601
```

As três primeiras instruções, encarregadas da música, usam a rotina **BEEPER** no endereço 949 da ROM. Os parâmetros da duração e da tonalidade são fornecidos pelo método usual, através dos pares de registros DE e HL.

O NÍVEL DO JOGO

O nível de dificuldade do jogo é carregado da variável correspondente, em 57334, para o acumulador. O conteúdo do acumulador é em seguida incrementado. Precisamos, porém, impedir que seu valor seja maior do que 3, já que existem só quatro níveis de dificuldade. Para isso, usamos a instrução **res 2, a** que ajusta com 0 o bit 2 do acumulador. Quando o valor deste chega a 4, o bit 2 é colocado em 0. O conteúdo do acumulador volta, assim, a 0, trazendo o jogo para o primeiro nível.

O resultado dessa operação é armazenado no endereço 57334, onde será utilizado ao se ajustar o jogo.

A VELOCIDADE

O valor do atraso do jogo — que fica no endereço 58732 — é carregado no acumulador, decrementado e armazenado de volta em 58732. Aceleramos, assim, o jogo, uma vez que o tempo gasto pelo processador no laço de atraso da rotina principal diminui.

O atraso foi originalmente ajustado com 50. Como Willie não chegará à recompensa mais do que cinquenta vezes, o jogo se tornará cada vez mais rápido — e, portanto, mais difícil. Cada vez que você enfrentar os mesmos quatro níveis, eles estarão mais rápidos.

CONTAGEM DE PONTOS

Por ter alcançado o prêmio, Willie recebe mais 500 pontos. Para isso, chamamos a rotina do escore no endereço 58900 e fornecemos os parâmetros nos registros A e B.

O valor 2 colocado em A especifica o segundo dígito a partir da esquerda — o das centenas —, que será incrementado. O valor 5 em B informa à rotina o número de vezes que esse dígito deve ser aumentado. Incrementando as centenas cinco vezes, acrescentamos 500 pontos ao escore.

Em seguida, o processador volta para a rotina de nova vida — rotulada **nlv** —, em 58601, e coloca nosso personagem na base da encosta.

O PLACAR

A pequena rotina apresentada a seguir acerta o placar de Willie quando ele alcança uma recompensa ou escala uma outra parte da encosta.





- NÍVEL DE DIFICULDADE
- FUNDO MUSICAL
- ACELERAÇÃO DO RITMO DO JOGO
- WILLIE ALCANÇA
- O PRÊMIO
- CONTAGEM DE PONTOS
- EXIBIÇÃO DO PLACAR
- ALTERAÇÃO DOS DÍGITOS
- DE VOLTA AO SOPÉ

```
 10 REM org 59900
 20 REM scn ld ix,57337
 30 REM ld d,0
 40 REM ld e,a
 50 REM add ix,de
 60 REM scr push ix
 70 REM call sdi
 80 REM pop ix
 90 REM djnz scr
100 REM call 58939
110 REM ret
120 REM sdi ld a,(ix+0)
130 REM inc a
140 REM cp 10
150 REM jr nz,sno
160 REM ld a,0
170 REM ld (ix+0),a
180 REM dec ix
190 REM jr sdi
200 REM sno ld (ix+0),a
210 REM ret
```

A posição de memória 57337 corresponde ao início da variável do escore, que é carregada com 0 pela rotina de inicialização dada no artigo da página 995. O registro D é carregado com 0 e E, com o conteúdo do acumulador. Lembre-se de que o acumulador carrega o número de coluna quando o processador entra nesta rotina.

O conteúdo do par DE é adicionado ao conteúdo de IX e o resultado perma-

nece em IX. Isso faz com que o apontador de dados percorra os dígitos armazenados a partir de 57337, até chegar ao especificado pelo conteúdo de A. A posição é temporariamente armazenada, colocando-se o conteúdo de IX na pilha. A rotina **sdi** é chamada.

ALTERANDO OS DÍGITOS

A rotina **sdi** é a que trata dos dígitos. Ela começa carregando o acumulador com o dígito apontado por IX. O deslocamento do 0 é necessário aqui por causa do formato da instrução.

O dígito é então incrementado e comparado com 10. Se ele for 10, será preciso incrementar também no próximo dígito. Caso contrário — ou seja, se o primeiro dígito ainda não foi incrementado até 10 —, a instrução **jr nz** manda o processador para o rótulo **sno**, que volta a armazenar o dígito incrementado na posição de onde veio.

Se o dígito que você está alterando foi incrementado até 10, o salto não ocorre e o processador continua com a próxima instrução. O valor 0 é armazenado no dígito apropriado. Em seguida, IX é decrementado, passando a apontar para o dígito imediatamente à esquerda. A instrução **jr sdi** manda então o processador de volta para **sdi**, iniciando novamente a rotina de incremento no próximo dígito.

Se este foi incrementado até 10, o processador continua no laço. Mas, cedo ou tarde, um dígito que não chegou a 10 será encontrado. O processador irá então para **sno**, armazenando o último dígito e retornando para o lugar da rotina **scr** de onde **sdi** foi chamada.

MAIS PLACAR

Quando o processador retorna da rotina, o apontador IX é recuperado da pilha outra vez. Você pode agora perceber por que precisamos armazená-lo ali: se o dígito foi incrementado até 10, **sdi** terá deslocado o apontador IX para o dígito seguinte; caso você tente incrementá-lo mais uma vez, irá alterar o dígito errado.

O laço desta parte da rotina é fechado por uma instrução **djnz**, que decrementa o conteúdo de B até zerá-lo. Como você deve se lembrar, B carrega o número de vezes que o dígito do escore tem que ser incrementado quando o processador entra na rotina.

O processador continua no laço atualizando o escore o número de vezes inicialmente especificado por B. Quando





B chega a 0, a rotina de impressão em 58939 é chamada, imprimindo o novo escore na tela.

Em seguida, o processador retorna.



Esta pequena rotina executa a melodia da recompensa, coloca Willie no próximo nível, acelera o jogo e acrescenta pontos no placar.

```
10  ORG 20721
20  RWD LDA #255
30  LDX #150
40  JSR SOUND
50  LDA 18238
60  INCA
70  ANDA #3
80  STA 18238
90  DEC DLL+1
100 LDB #5
110 LDA #3
120 JSR SCI
130 LBRA NLV
140 SCI EXG A,B
150 SCT LDX #18240
160 ABX
170 PSHS A,X
180 JSR SDI
190 PULS A,X
200 DECA
210 BNE SCT
220 JSR PRSC
230 RTS
240 SDI LDA,X
250 INCA
260 CMPA #10
270 BNE SNO
280 CLR ,X
290 LEAX -1,X
300 BRA SDI
310 SNO STA ,X
320 RTS
330 SOUND PSHS A
340 LDA $FF01
350 ANDA #247
360 STA $FF01
370 LDA $FF03
380 ANDA #247
390 STA $FF03
400 LDA $FF23
410 ORA #8
420 STA $FF23
430 ORCC #$50
440 PULS A
450 PSHS X
460 LDB #252
470 SBN STB $FF20
480 SC LEAX -1,X
490 BNE SC
500 LDX,S
510 CLR $FF20
520 SD LEAX -1,X
530 BNE SD
540 LDX ,S
550 DECA
560 BNE SBN
570 ANDCC #$AF
580 PULS X
590 RTS
600 CLICK LDX #98
610 LDA #4
620 JSR SOUND
630 RTS
640 DLL EQU $51ED
650 NLV EQU $48F7
660 PRSC EQU $4C77
```

As três primeiras instruções, encarregadas da música, usam a rotina **SOUND**, chamada pela linha 40. Os parâmetros da duração e da tonalidade são fornecidos pelo método usual, através de números carregados em A e X.

NÍVEL DE DIFICULDADE

A é carregado com o conteúdo de 18238, a posição de armazenamento do nível de dificuldade. Esse valor é incrementado e a operação AND é feita com o número 3. Apagamos, assim, os seis bits mais significativos e evitamos que o nível do jogo ultrapasse o valor 3. O resultado é armazenado de volta na posição 18238.

Em seguida, a variável na posição de memória $51EE é decrementada e o jogo se torna um pouco mais rápido.

Para atualizar o escore, carrega-se 5 em B e 3 em A; a rotina **SCI**, fornecida a seguir, é então chamada. B carrega o número de vezes que o dígito será incrementado e A identifica esse dígito. Desta vez, portanto, acrescentamos 500 pontos ao escore.

Finalmente, o processador volta para a rotina **NLV**, que irá colocar o próximo nível na tela.

CÁLCULO DO PLACAR

Para serem utilizados na próxima rotina, os conteúdos dos registradores A e B precisam ser trocados. Isso é feito pela instrução **EXG**. X é então carregado com 18240, o endereço inicial dos valores do escore.

ABX adiciona a X o conteúdo de B — que é o número do dígito a ser incrementado. Em outras palavras, essa instrução desloca o apontador em X, que estava no início dos dados do escore, para a posição do dígito que se pretende incrementar. O conteúdo de A (o número de vezes que o dígito será incrementado) e o de X (a posição de memória desse dígito) são colocados na pilha de máquina. Em seguida, o processador salta para a sub-rotina **SDI** que executa o incremento. Os conteúdos de A e X são recuperados da pilha.

A é decrementado e, se não tiver chegado a 0, a instrução **BNE SCT** manda o processador continuar no laço. A rotina de incremento do dígito é executada A vezes, incrementando o dígito apropriado a cada volta do laço.

Com A reduzido a 0 e o escore acertado, o processador sai do laço e vai para a rotina **PRSC**, que imprime o resultado na tela. Depois disso, o processador encontra **RTS** e retorna para a rotina principal.

ALTERANDO OS DÍGITOS

A é carregado com o conteúdo da posição de memória apontada por X. Este corresponde ao número contido no dígito apropriado do escore.

O número é incrementado e o resultado é comparado com 10. Se ainda não,

for igual a 10, a instrução **BNE** coloca o processador no rótulo **SNO**, que volta a armazenar o dígito que foi incrementado no endereço apontado por X. O processador retorna.

Se o resultado for igual a 10, o dígito apontado por X é ajustado com 0 — ou seja, é limpo. X é então decrementado, fazendo o apontador se mover para o dígito imediatamente à esquerda.

BRA SDI manda o processador de novo para o laço de incremento de dígito. Depois que o próximo dígito é incrementado, verifica-se se ele ultrapassou o valor 9. Encontrando um dígito que não exceda o valor máximo, o processador o armazena na posição apontada por X. Observe que, ao voltar da rotina de escore, o processador recupera o valor de X da pilha.

A rotina **SOUND** trabalha exatamente do mesmo modo que a rotina encarregada de executar *Greenleaves*. Porém, toca apenas uma vez e usa a pilha como fonte de dados.

Depois de **SOUND**, há uma outra pequena rotina chamada **CLICK**. Ela executa a trilha sonora da caminhada de Willie, carregando os parâmetros em X e A e chamando a rotina **SOUND**.

Para testar a rotina, use as teclas M e N e este programa em BASIC.

```
10 POKE 3000,S7
20 EXEC 19426
30 EXEC 19902
40 GOTO 30
```



Esta pequena rotina coloca Willie no nível seguinte, aumenta a velocidade do jogo e acrescenta os pontos ganhos ao placar.

```
org 55506
ld a,(-5228)
inc a
res 2,a
ld (-5228),a
ld a,(54133)
dec a
ld (54133),a
ld a,2
 ld b,5
 call 55532
 jp 53888
```

O NÍVEL DE DIFICULDADE

O nível de dificuldade do jogo é transferido da variável equivalente em −5228 para o acumulador. A seguir, o conteúdo desse é incrementado. Como existem apenas quatro níveis no jogo, o valor dessa variável não deve ultrapassar 3. Para evitar que isso ocorra, utilizamos a instrução **res 2,a**, que ajusta com 0 o bit 2 do acumulador. Assim, quando o valor incrementado chega a 4, o bit 2 é ajustado com 0. O conteúdo do acumulador volta, então, a 0, colocando o jogo novamente no primeiro nível.

O resultado dessa operação é armazenado em −5228, onde o novo valor será utilizado no controle do jogo.

VELOCIDADE

O atraso do jogo em 54133 é colocado no acumulador, decrementado e armazenado de volta em 54133. Isso acelera o jogo, já que o tempo gasto pelo processador no laço de atraso da rotina principal diminui.

Como você poderá verificar na última rotina da série *Avalanche*, esse atraso foi originalmente ajustado com 50. Assim, sempre que aumenta o número de pontos no placar, o jogo se torna mais rápido e difícil. Cada vez que você enfrentar os mesmos quatro níveis, a velocidade será maior.

CONTAGEM DOS PONTOS

Por ter alcançado o prêmio, Willie recebe mais 500 pontos. Para isso, chamamos a rotina de contagem e fornecemos os parâmetros nos registros A e B. O valor 2 colocado em A especifica o segundo dígito a partir da esquerda — o das centenas —, que será incrementado. O valor 5 em B informa à rotina o número de vezes que esse dígito deve ser aumentado. Incrementando as centenas cinco vezes, acrescentamos 500 pontos ao escore. Em seguida, o processador vai para a rotina principal do jogo, em 53888, e coloca Willie na base da encosta.

O PLACAR

A pequena rotina que se segue acerta os pontos de Willie quando ele alcança uma recompensa ou escala uma parte da encosta.

```
org 55532
po ld hl,-5219
ld d,0
ld e,a
add hl,de
pd push hl
call sd
pop hl
djnz pd
call 54023
ret
sd ld a,(hl)
inc a
cp 10
jr nz,sn
ld a,0
ld (hl),a
dec hl
jr sd
sn ld (hl),a
ret
end
```

A posição de memória −5219 corresponde ao endereço inicial das variáveis que contêm os dígitos do escore, carregadas com 0 pela rotina de inicialização dada em artigo anterior. O registro D é carregado com 0 e o registro E, com o conteúdo do acumulador. Lembre-se de que o acumulador carrega o número do dígito quando o processador entra nesta rotina.

O conteúdo do par DE é somado em HL. A operação faz o apontador de dígitos percorrer as variáveis onde seus valores estão armazenados, a partir de −5219, até o dígito indicado pelo conteúdo de A. O conteúdo de HL é colocado na pilha, onde ficará temporariamente armazenado, e **sd** é chamada.

ALTERANDO OS DÍGITOS

A rotina **sd**, que trata dos dígitos, começa carregando o acumulador com o



valor do dígito apontado por HL. Este é incrementado e comparado com 10. Se for igual a 10 — ou seja, se ultrapassou o maior valor de um dígito decimal, 9 —, será preciso incrementar o próximo dígito. Se o dígito analisado ainda não ultrapassou 9, a instrução **jr nz** manda o processador para o rótulo **sn**, que armazena seu valor na variável correspondente.

Quando o dígito que está sendo alterado chega a 10, o salto não ocorre e o processador continua com a próxima instrução. O valor 0 é armazenado nesse dígito e o par HL é decrementado, passando a apontar para o dígito imediatamente à esquerda. A instrução **jr sd** manda o processador de volta para **sd** e a rotina de incremento recomeça para o próximo dígito.

Se este também já ultrapassou 9, o processador continua no laço. Mas, cedo ou tarde, um dígito que não chegou a 10 será encontrado. O processador irá então para **sn**, armazenando o último dígito e retornando para o lugar da rotina **pd** de onde **sd** foi chamada.

MAIS PLACAR

Quando o processador retorna da rotina, o apontador HL é recuperado da pilha. Você pode agora perceber por que precisamos armazená-lo ali: se algum dígito foi incrementado até 10, a rotina **sd** terá deslocado o apontador. Caso não tivéssemos armazenado o valor inicial na pilha, desta vez estaríamos alterando o dígito errado.

O laço é fechado por uma instrução **djnz**, que decrementa o conteúdo de B até zerar esse registro. Como você deve se lembrar, B carrega o número de vezes que o dígito do escore tem que ser incrementado quando o processador entra nesta rotina.

O processador continua no laço **pd** atualizando o escore o número de vezes inicialmente especificado por B. Quando B chega a 0, a rotina de impressão dos dígitos em 54023 é chamada, imprimindo o novo escore na tela.

Em seguida, o processador retorna.



OS DADOS VÃO ROLAR

INPUT apresenta, finalmente, um jogo de computador projetado para vários participantes. Reúna a família e os seus amigos e role os dados nesta brincadeira de sorte e de perícia.

Até aqui, a maioria dos jogos de *INPUT* — sejam os de aventura, estratégia ou videogame — basearam-se no confronto computador-usuário. A maior dificuldade nesse tipo de disputa estava, como vimos, na criação de regras e situações que simulassem desafio ou, no caso dos jogos de estratégia, na transformação do computador em um adversário inteligente e de bom nível. O jogo que apresentamos neste artigo é diferente: o confronto pode se dar entre até seis jogadores. O computador não participa como adversário - cabe-lhe apenas conferir as regras e manter o placar. A máquina evita, assim, que alguém trapaceie, e, o que é melhor, libera os jogadores da contagem do placar, possibilitando que se concentrem só nas questões estratégicas.

Este jogo não passa de uma versão computadorizada do famoso *Yatch*, ou pôquer com dados. Combinando sorte e estratégia, é muito absorvente, mas suas regras são simples, parecidas com as do pôquer. Cada participante tem direito a jogar cinco dados de uma só vez (no nosso caso, o computador simula os dados rolando). Se o jogador não estiver satisfeito com o resultado obtido, poderá lançar os dados mais duas vezes (num total de três jogadas). Na segunda e terceira jogadas, é permitido escolher quais dentre os cinco dados serão lançados novamente, para que se consiga a melhor "mão" possível. Depois do terceiro lançamento, o jogador deve indicar o grupo do placar no qual sua combinação de dados vai ficar (cada grupo poderá ser usado apenas uma vez). A partida continua, então, com o próximo jogador.

Os grupos de placar são:

GRUPO	PONTOS
UM	soma dos 1 obtidos
DOIS	soma dos 2 obtidos
.	
.	
.	.
SEIS	soma dos 6 obtidos
FULL HOUSE	soma dos cinco dados
CURTO	15
LONGO	30
MISTO	soma dos cinco dados
YATCH	50

Um CURTO é uma combinação de quatro dados em seqüência (1,2,3,4 ou 2,3,4,5) e um LONGO, uma combinação de cinco dados também em seqüência (1,2,3,4,5 ou 2,3,4,5,6). Um FULL HOUSE compõe-se de uma trinca e um par (5,5,5,1,1, por exemplo), enquanto um MISTO, como seu próprio nome diz, aceita qualquer tipo de combinação de dados. O YATCH, finalmente, é o grupo formado por cinco dados iguais.

O resultado obtido numa jogada deve ser colocado num dos grupos. Lembre-se de que só se pode indicar cada grupo uma vez. Para selecionar o grupo e executar a escolha usam-se as setas e a barra de espaço.

Muitas vezes não é possível obter uma combinação de dados que satisfaça um dos grupos vazios (os grupos cheios não podem ser utilizados novamente). Neste caso, a única opção é sacrificar um dos grupos vazios — de preferência, um dos que rendem menos pontos —, eliminando-o. A melhor estratégia, portanto, é preencher primeiro os grupos que rendem mais pontos — que são também os mais difíceis — evitando-se que venham a ser sacrificados.

O programa está dividido em três partes principais: rotina de inicialização, laço-mestre e rotinas chamadas pelo laço-mestre.

INICIALIZAÇÃO

Esta parte do programa configura os UDG (gráficos definidos pelo usuário) que representarão os dados na tela (menos nos micros das linhas Apple e TK-2000), inicializa as variáveis e armazena os nomes dos jogadores.



```
10 CLS:X$=CHR$(13)+CHR$(10):DIM
D$(6.4)
20 FORK=1TO6·FORJ=1TO3:FORL=1TO
3:READA:D$(K.J)=D$(K.J)+CHR$(1)
+CHR$(219-145*A):NEXT:NEXT:NEXT
30 DATA 0.0.0.0,1,0.0.0.0
40 DATA 1,0,0,0,0,0,0,0,1
50 DATA 1,0.0.0.1.0.0.0.1
60 DATA 1,0,1,0,0.0.1,0,1
70 DATA 1,0,1,0,1.0.1,0,1
80 DATA 1.0.1,1,0.1.1.0,1
85 LOCATE7.10:PRINT"Quantos jog
adores (1-6)?";
90 A$=INKEY$:IFA$<"1" OR A$>"6"
THEN90
100 PRINTA$:NP=VAL(A$):CLS
110 FORN=1TONP:PRINT"nome do jo
gador";N;:INPUTN$(N):NEXT
120 CLS:CL$=STRING$(35," ")
130 DIMO(NP.12),P(NP.12),S(NP.1
0)
```



```
10  HOME : DIM A$(12)
20 CL$ = "

30  FOR T = 1 TO 6: READ A$:DC$
(T) = A$· NEXT
40  FOR T = 1 TO 12
50  READ A$:A$(T) = A$ +  LEFT$
 (".............",12 -  LEN (A$)
): NEXT
60  VTAB (10): PRINT  TAB( 10);
"QUANTOS JOGADORES (1-6)?";
70  GET A$: IF A$ < "1" OR A$ >
 "6" THEN 70
80 NP =  VAL (A$): PRINT NP: DI
M N$(NP).SC(NP).O(NP.12),P(NP.1
2),S(NP.10)
90  HOME : FOR N = 1 TO NP: PRI
NT  TAB( 1);"NOME DO JOGADOR ";
N;" = ";
100  INPUT N$(N): NEXT
1170  DATA  <1>,<2>,<3>
1180  DATA  <4>.<5>,<6>
1190  DATA   UNS,DOIS.TRES,QUAT
ROS,CINCOS,SEIS,4 IGUAIS
1200  DATA   CASA CHEIA,CURTO,L
ONGO,MISTO,YATCH
```



```
20 LET Q$="...........": LET
Z$="             ": DIM C(13):
FOR N=1 TO 13: READ C(N):
NEXT N: DIM T(5): DIM R(5):
DIM D(5)
30 FOR N=USR "A" TO USR "G"+7
: READ A: POKE N,A: NEXT N
40 DATA 2,3,4,5.6,7,11,14,17,
20,23,25,27
50 DATA 0.0.0.24,24,0,0,0
60 DATA 0,6.6,0,0.96,96.0
70 DATA 3.3,0.24,24.0,192,192
80 DATA 0.102,102.0,0,102,102
,0
90 DATA 195,195,0.24.24.0,195
,195
```

```
100 DATA 102,102,0,102,102,0,
102,102
110 DATA 0,24,48,96,255,96,48,
24
120 PRINT AT 10,13;"YATCH":
INK 1: PRINT AT 12,7;"QUANTOS
JOGADORES"''TAB 11;"(1 A 6)"
130 INPUT NP: LET NP=INT (NP):
IF NP<1 OR NP>6 THEN  GOTO 130
140 DIM O(NP,12): DIM P(NP,12)
: DIM S(NP,5): DIM N$(NP,6):
DIM Q(NP)
150 FOR N=1 TO NP: CLS : PRINT
AT 8,5;"JOGADOR ";(N);"."''TAB
5;"QUAL E SEU NOME ?": INPUT
W$: IF LEN W$>6 THEN  LET W$=
W$( TO 6)
160 LET N$(N)=Z$( TO 3-(LEN W$
)/2)+W$: NEXT N
```

```
10 CLS:X$=CHR$(13):DIM D$(6,4)
20 FOR K=1 TO 6:FOR J=1 TO 3:FO
R L=1 TO 3:READ A:D$(K,J)=D$(K,
J)+CHR$(128+65*A):NEXT
30 D$(K,J)=D$(K,J)+CHR$(133):NE
XT
40 D$(K,J)=STRING$(3,131)+CHR$(
135):NEXT
50 DATA 0,0,0,0,1,0,0,0,0,1,0,0
,0,0,0,0,0,1
60 DATA 1,0,0,0,1,0,0,0,1,1,0,1
,0,0,0,1,0,1
70 DATA 1,0,1,0,1,0,1,0,1,1,0,1
,1,0,1,1,0,1
80 PRINT:PRINT"QUANTOS JOGADORE
S (1-6) ?";
90 A$=INKEY$:IF A$<"1" OR A$>"6
" THEN 90
100 PRINT A$:NP=VAL(A$):CLS
110 FOR N=1 TO NP:PRINT @65,"JO
GADOR";N:PRINT" QUAL E SEU NOME
 ?":INPUT N$(N)
120 CLS:NEXT
130 DIM O(NP,12),P(NP,12),S(NP,
10)
```

O LAÇO-MESTRE

A estrutura do jogo é bem simples, e consiste nestas poucas linhas:

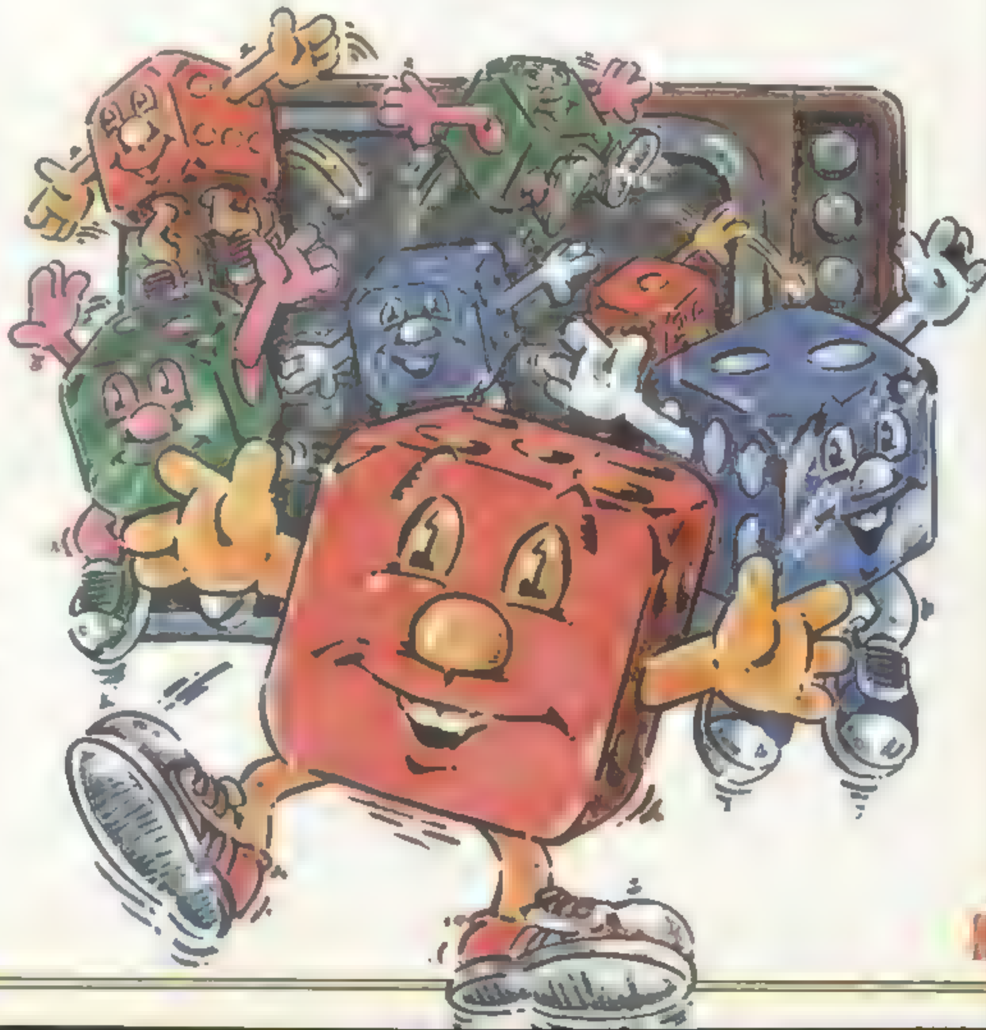
```
140 FORR=1TO5:FORI=1TO12:FORN=1
TONP
150 CLS:Y=2:GOSUB980:GOSUB190
170 CLS:GOSUB350:CLS:NEXTN,I
180 CLS:GOSUB990:NEXTR:END
980 LOCATE19-LEN(N$(N))/2,Y:PRI
NTN$(N):RETURN
```



```
140  FOR R = 1 TO 5: FOR I = 1
TO 12: FOR N = 1 TO NP
150  HOME :Y = 2: GOSUB 980: GO
SUB 190
170  HOME : GOSUB 350: HOME : N
EXT N,I
180  HOME : GOSUB 990: NEXT R:
END
980  HTAB (19 -  LEN (N$(N)) /
2): VTAB (Y): PRINT N$(N): RETU
RN
```

```
170 FOR R=1 TO 5: FOR I=1 TO
12: FOR N=1 TO NP
180 BORDER 4: INK 0: PAPER 4:
CLS : PRINT AT 3,13;N$(N)
190 FOR M=5 TO 27: PRINT
PAPER 0;AT 5,M;" ";AT 19,M;" "
: NEXT M
200 FOR M=6 TO 18: PRINT
PAPER 0;AT M,5;" ";AT M,27;" "
: NEXT M
210 GOSUB 240: PAUSE 0: GOSUB
 430
230 NEXT N: NEXT I: GOSUB 1290
: NEXT R: STOP
```

```
140 FOR R=1 TO 5:FOR I=1 TO 12:
FOR N=1 TO NP
```





```
150 CLS:W=6:Y=2:GOSUB 980:GOSUB.
 190
160 SOUND 50,3:FOR E=1 TO 800:N
EXT
170 CLS:GOSUB 350:CLS:NEXT N,I
180 CLS·GOSUB 990:NEXT R:END
9870 PRINT @Y*32+W-((LEN(N$(N))
)/2),N$(N):RETURN
```

O laço-mestre compõe-se de três laços encadeados (um dentro do outro) que controlam o jogo. **R** é o número da partida; **I**, o número de jogadas dentro de uma partida e **N**, o número de jogadores. As rotinas chamadas por esse laço fazem os dados rolar e apresentam os placares parcial e final. Todos os micros, menos o Spectrum, chamam uma pequena rotina para centralizar os nomes na tela. As rotinas chamadas pelo laço-mestre também são subdivididas, como veremos a seguir.

OS DADOS VÃO ROLAR

A primeira dessas rotinas rola os dados, exibe-os na tela e chama duas outras rotinas.

Acrescente as linhas que se seguem à parte do programa já digitada:



```
190 T=1:FORD=1TO5:T(D)=INT(RND(
-TIME)*6)+1:NEXT
200 LOCATE0,T*2+3:PRINT"JOGO";T
.
210 GOSUB970:IFT=3 THEN 310
220 C=1·FOR D=1 TO 5
230 LOCATE9+D*4,9:PRINT"?"
240 A$=INKEY$·IFA$<>"N"ANDA$<>"
n"ANDA$<>"s"ANDA$<>"S"THEN240
250 IF A$="N"ORA$="n"THENPLAY"O
3L64CB":GOTO270
260 PLAY"O6L64CB":TR(C)=T(D)·C=
C+1
270 LOCATE0,9·PRINTCL$·NEXTD
280 IFC=6THENGOSUB340:RETURN
290 FORD=C TO 5:TR(D)=INT(RND(-
TIME)*6)+1:NEXTD
300 GOSUB340
310 T=T+1
320 IFT<>4THEN200
330 FORE=1TO700:NEXT·RETURN
340 FORD=1TO5:T(D)=TR(D):NEXTD:
RETURN
970 FORD=1TO5:FORG=1TO4:LOCATED
*4+8,G+4:PRINTD$(T(D),G);:NEXTG
,D:RETURN
```



```
190 T = 1: FOR D = 1 TO 5:T(D)
= INT ( RND (1) * 6) + 1: NEXT
200  VTAB (T * 2 + 3): PRINT "J
OGO ";T;
210  GOSUB 970: IF T = 3 THEN 3
10
220 C = 1: FOR D = 1 TO 5
230  HTAB (9 + D * 4): VTAB (9)
: PRINT "?";
240  GET A$: IF A$ <  > "S" AND
 A$ <  > "N" THEN 240
250  IF A$ = "N" THEN 270
260 TR(C) = T(D):C = C + 1
270  HTAB (9 + D * 4)· VTAB (9)
: PRINT " ": NEXT D
280  IF C = 6 THEN  GOSUB 340:
RETURN
290  FOR D = C TO 5:TR(D) =  IN
T ( RND (1) * 6) + 1: NEXT
300  GOSUB 340
310 T = T + 1
320  IF T <  > 4 THEN 200
330  FOR E = 1 TO 700: NEXT : R
ETURN
340  FOR D = 1 TO 5:T(D) = TR(D
): NEXT : RETURN
970  FOR D = 1 TO 5: HTAB (D *
4 + 8): VTAB (6): PRINT DC$(T(D
));: NEXT : RETURN
```



```
240 LET T=1: FOR D=1 TO 5: LET
T(D)=INT (RND*6)+1: NEXT D
250 PRINT AT 6+T*3,7;"JOGO ";T
260 GOSUB 1180
270 IF T=3 THEN  GOTO 390
280 LET C=1: FOR D=1 TO 5
290 PRINT AT 7+T*3,16+D*2;"?"
300 FOR J=1 TO 50: NEXT J
310 LET A$=INKEY$: IF A$=""
THEN  GOTO 310
320 IF A$="N" THEN  SOUND .1,
10: GOTO 360
330 IF A$<>"S" THEN  GOTO 310
340 SOUND .1,30
350 LET R(C)=T(D): LET C=C+1
360 PRINT AT 7+T*3,16+D*2;" ":
NEXT D
370 IF C=6 THEN  GOSUB 420:
LET T=4: GOTO 400
380 FOR D=C TO 5: LET R(D)=INT
(RND*6)+1: NEXT D: GOSUB 420
390 LET T=T+1
400 IF T<>4 THEN  GOTO 250
410 RETURN
420 FOR D=1 TO 5: LET T(D)=R(D
): NEXT D: RETURN
1180 FOR D=1 TO 5: PRINT  PAPER
 2: INK 6: BRIGHT 1;AT 6+T*3,16
+D*2;CHR$ (143+T(D)): PAUSE 2:
SOUND .01,RND*40: NEXT D: RETUR
N
```



```
140 FOR R=1 TO 5:FOR I=1 TO 12:
FOR N=1 TO NP
150 CLS:W=6:Y=2:GOSUB 980:GOSUB
 190
160 SOUND 50,3:FOR E=1 TO 800:N
EXT
170 CLS:GOSUB 350:CLS:NEXT N,I
180 CLS:GOSUB 990:NEXT R:END
190 T=1:FOR D=1 TO 5:T(D)=RND(6
):NEXT
200 PRINT @64*T+64,"LANCE:";T;
210 GOSUB 970:IF T=3 THEN 310
220 C=1:FOR D=1 TO 5
230 PRINT @288,TAB(9+D*4)"?"
240 A$=INKEY$:IF A$<>"N" AND A$
<>"S" THEN 240
250 IF A$="N" THEN SOUND 10,1:G
OTO 270
260 SOUND 100,1:TR(C)=T(D):C=C+
1
270 NEXT D:PRINT @288
280 IF C=6 GOSUB 340:RETURN
290 FOR D=C TO 5:TR(D)=RND(6):N
EXT D
300 GOSUB 340
310 T=T+1
320 IF T<>4 THEN 200
330 RETURN
340 FOR D=1 TO 5:T(D)=TR(D):NEX
T D:RETURN
970 FOR D=1 TO 5:FOR G=1 TO 4:P
RINT @136+G*32+D*4,D$(T(D),G);:
NEXT G,D:RETURN
9870 PRINT @Y*32+W-((LEN(N$(N))
)/2),N$(N):RETURN
```

No primeiro lançamento, utilizam-se os cinco dados. O resultado é exibido na tela por meio da rotina da linha 970 (1180, no Spectrum). O jogador escolhe então quais são os dados "bons" e quais são os "ruins", indicando os primeiros com um S e os segundos com um N. Ele terá mais duas chances de lançar os dados ruins, devendo teclar S(im) para os dados que quer "segurar" e N(ão) para os demais.

Os números dos cinco dados iniciais são armazenados na matriz T( ). Depois do primeiro lançamento, os dados que "seguramos" são guardados temporariamente na matriz R( ), que é preenchida com outros valores randômicos até ficar com cinco números. Esses valores são levados de volta para a matriz T( ) por meio da rotina da linha 340 (420, no Spectrum). Os resultados são exibidos na tela mais uma vez e o processo se repete no próximo lançamento.

O PLACAR

A maior parte da rotina dedica-se aos cálculos do placar e à verificação de entradas.

```
350 CLS
360 Y=0:GOSUB980:PRINTSTRING$(3
0,205);" PLACAR"
370 PRINT"UNS..........:"X$"DOIS
.........:"X$"TRES.........:"X$"Q
UATROS.....:"X$"CINCOS......:"X
$"SEIS........:"
380 PRINT"4 IGUAIS....:"X$"CASA
 CHEIA..:"X$"CURTO.......:"X$"L
ONGO.......:"X$"MISTO.......:"X
$"YATCH.......:"X$
390 PRINT"TOTAL.......:";
400 GOSUB460
410 GOSUB490:GOSUB460
420 LOCATE22,5:PRINT"<qualquer
tecla>"
440 A$=INKEY$:IFA$=""THEN440
450 RETURN
460 FORD=1TO12:IFP(N,D)=1 THENL
OCATE16,1+D:PRINT"X";
470 IFO(N,D)<>0THENLOCATE15,1+D
:PRINTO(N,D);
480 NEXTD:C=0:FORD=1TO12:C=C+O(
N,D):NEXT:LOCATE15,15:PRINTC;:R
ETURN
490 LOCATE2,20:PRINT"TEMPO"R;:L
OCATE1,21:PRINT"JOGADA"I;
500 LOCATE0,17:PRINT"RESULTADO
=";
510 FORD=1TO5:FORG=1TO4:LOCATE9
+4*D,15+G:PRINTD$(T(D),G);:NEXT
G,D
520 LOCATE16,20:PRINT"ESCOLHA O
 GRUPO";
530 A=1
540 LOCATE13,1+A:PRINTCHR$(60);
550 B$=INKEY$:IFB$<>" "ANDB$<>C
HR$(30)ANDB$<>CHR$(31)THEN550
560 IFB$=" "THEN620
570 LOCATE13,1+A:PRINT" ";
580 IFB$=CHR$(30)ANDA>1THENA=A-
1
590 IFB$=CHR$(31)ANDA<12THENA=A
+1
600 PLAY"O6A"
610 GOTO540
620 LOCATE13,1+A:PRINT" ";:IFP(
N,A)<>0THEN950
630 IFA>6THEN700
640 C=0
650 FORD=1TO5:IFT(D)=A THENC=C+
1
660 NEXTD
670 O(N,A)=C*A
680 P(N,A)=1
690 RETURN
700 IFA=11THENFORD=1TO5:O(N,11)
=O(N,11)+T(D):NEXT:P(N,11)=1:RE
TURN
710 FORD=1TO5:D(D)=0:NEXT:B=0:F
ORE=1TO6:C=0:FORD=1TO5:IFT(D)=E
 THENC=C+1
720 NEXTD:IFC<>0THENB=B+1
730 NEXTE
740 G=1:FORF=1TO6:GOSUB1140:IFC
<>0THEND(G)=F:G=G+1
750 NEXTF
760 P(N,A)=1:A=A-6:ONA GOTO770,
810,830,870,960,890
770 IFB>2THEN900:IFB=1GOSUB1160
:O(N,7)=C*4:RETURN
780 F=1
790 GOSUB1140:F=F+1:IFC<>4ANDF<
>7THEN790ELSEIFC<4THEN900
800 O(N,7)=4*(F-1):RETURN
810 IFB<>2THEN900ELSEF=D(1):GOS
UB1140:IFC=3THEN820ELSEF=D(2):G
OSUB1140:IFC<>3THEN900
820 FORD=1TO5:O(N,8)=O(N,8)+T(D
):NEXT:RETURN
830 IFB<>4THEN850ELSEGOSUB1160:
IFC<>18ANDC<>10ANDC<>14 OR(C=14
ANDD(4)=6)THEN900
840 O(N,9)=15:RETURN
850 IFB<>5THEN900ELSEGOSUB1160:
IFC<>20ANDC<>15ANDC<>16ANDC<>19
THEN900
860 GOTO840
870 IFB<>5THEN900ELSEGOSUB1160:
IFC<>20ANDC<>15THEN900
880 O(N,10)=30:RETURN
890 IFB<>1THEN900ELSEO(N,12)=50
:PLAY"GFG":FORE=1TO700:NEXT:RET
URN
900 PLAY"O3GGG":LOCATE22,5:PRIN
T"ilegal  ELIMINA?";
910 A$=INKEY$:IFA$<>"S"ANDA$<>"
s"ANDA$<>"N"ANDA$<>"n"THEN910
920 LOCATE22,5:PRINTSTRING$(17,
32);
930 IFA$="N"ORA$="n"THENP(N,A+6
)=0:GOTO530
940 P(N,A+6)=1:RETURN
950 PLAY"O3CFD":LOCATE22,5:PRIN
T"GRUPO OCUPADO";:FORE=1TO700:N
EXT:LOCATE22,5:PRINTSTRING$(14,
32);:GOTO530
960 RETURN
1140 C=0:FORD=1TO5:IFT(D)=F THE
NC=C+1
1150 NEXTD:RETURN
1160 C=0:FORD=1TOB:C=C+D(D):NEX
TD:RETURN
```

```
350  HOME
360  Y = 1: GOSUB 980: INVERSE :
 PRINT ">>>>>>>>>>>>>>>>>>>>>
>>> PLACAR <<<"
370  NORMAL : PRINT : FOR T = 1
 TO 12: PRINT A$(T): NEXT
380  PRINT : INVERSE : PRINT "T
OTAL.......": NORMAL
400  GOSUB 460
410  GOSUB 490: GOSUB 460
420  HTAB (22): VTAB (8): PRINT
 "<QUALQUER TECLA>";
440  GET A$: IF A$ = "" THEN 44
0
450  RETURN
460  FOR D = 1 TO 12: IF P(N,D)
 = 1 THEN  HTAB (15): VTAB (3 +
 D): PRINT "X";
470  IF O(N,D) <  > 0 THEN  HTA
B (15): VTAB (3 + D): PRINT O(N
,D);
```

```
480  NEXT D:C = 0: FOR D = 1 TO
 12:C = C + O(N,D): NEXT
485  HTAB (15): VTAB (17): PRIN
T C: RETURN
490  VTAB (22): PRINT  TAB( 3);
"TEMPO";R: PRINT  TAB( 1);"JOGA
DA ";I
500  VTAB (19): PRINT "RESULTAD
O = ";
510  FOR D = 1 TO 5: VTAB (19):
 HTAB (9 + 4 * D): PRINT DC$(T(
D)):: NEXT D
520  HTAB (16): VTAB (21): PRIN
T "ESCOLHA O GRUPO";
530 A = 1
540  HTAB (13): VTAB (3 + A): P
RINT  CHR$ (60):
550  GET B$: IF B$ < > CHR$ (
81) AND B$ < > CHR$ (90) AND
B$ < > CHR$ (32) THEN 550
560  IF B$ =  CHR$ (32) THEN 62
0
570  HTAB (13): VTAB (3 + A): P
RINT " ";
580  IF B$ =  CHR$ (81) AND A >
 1 THEN A = A - 1
590  IF B$ =  CHR$ (90) AND A <
 12 THEN A = A + 1
600  PRINT  CHR$ (7)
610  GOTO 540
620  HTAB (13): VTAB (3 + A): P
RINT " ";: IF P(N,A) < > 0 THE
N 950
630  IF A > 6 THEN 700
640 C = 0
650  FOR D = 1 TO 5: IF (T(D) =
 A) THEN C = C + 1
660  NEXT D
670 O(N,A) = C * A
680 P(N,A) = 1
690  RETURN
700  IF A = 11 THEN  FOR D = 1
TO 5:O(N,11) = O(N,11) + T(D):
NEXT :P(N,11) = 1: RETURN
710  FOR D = 1 TO 5:D(D) = 0: N
EXT :B = 0: FOR E = 1 TO 6:C =
0: FOR D = 1 TO 5: IF T(D) = E
THEN C = C + 1
720  NEXT D: IF C < > 0 THEN B
 = B + 1
730  NEXT E
740 G = 1: FOR F = 1 TO 6: GOSU
B 1140: IF C < > 0 THEN D(G) =
 F:G = G + 1
750  NEXT F
760 P(N,A) = 1:A = A - 6: ON A
GOTO 770,810,830,870,960,890
770  IF B > 2 THEN 900: IF B =
 1 GOSUB 1160:O(N,7) = C * 4: RE
TURN
780 F = 1
790  GOSUB 1140:F = F + 1: IF C
 < > 4 AND F < > 7 THEN 790
795  IF C < 4 THEN 900
800 O(N,7) = 4 * (F - 1): RETUR
N
810  IF B < > 2 THEN 900
812 F = D(1): GOSUB 1140: IF C
 = 3 THEN 820
814 F = D(2): GOSUB 1140: IF C
 < > 3 THEN 900
820  FOR D = 1 TO 5:O(N,8) = O(
N,8) + T(D): NEXT : RETURN
```



```
830  IF B < > 4 THEN 850
835  GOSUB 1160: IF C < > 18 A
ND C < > 10 AND C < > 14 OR (
C = 14 AND D(4) = 6) THEN 900
840 O(N,9) = 15: RETURN
850  IF B < > 5 THEN 900
855  GOSUB 1160: IF C < > 20 A
ND C < > 15 AND C < > 16 AND
C < > 19 THEN 900
860  GOTO 840
870  IF B < > 5 THEN 900
875  GOSUB 1160: IF C < > 20 A
ND C < > 15 THEN 900
880 O(N,10) = 30: RETURN
890  IF B < > 1 THEN 900
895 O(N,12) = 50: PRINT  CHR$ (
7): FOR E = 1 TO 700: NEXT : RE
TURN
900  PRINT  CHR$ (7); CHR$ (7):
 HTAB (22): VTAB (5): PRINT "il
egal. ELIMINA?";
910  GET A$: IF A$ < > "S" AND
A$ < > "N" THEN 910
920  HTAB (22): VTAB (5): PRINT
CL$;
930  IF A$ = "N" THEN P(N,A + 6
) = 0: GOTO 530
940 P(N,A + 6) = 1: RETURN
950  PRINT  CHR$ (7); CHR$ (7):
 HTAB (22): VTAB (5): PRINT "GR
UPO OCUPADO";: FOR E = 1 TO 700
: NEXT
955  HTAB (22): VTAB (5): PRINT
 CL$;: GOTO 530
960  RETURN
1140 C = 0: FOR D = 1 TO 5: IF
T(D) = F THEN C = C + 1
1150  NEXT D: RETURN
1160 C = 0: FOR D = 1 TO B:C =
C + D(D): NEXT D: RETURN
```

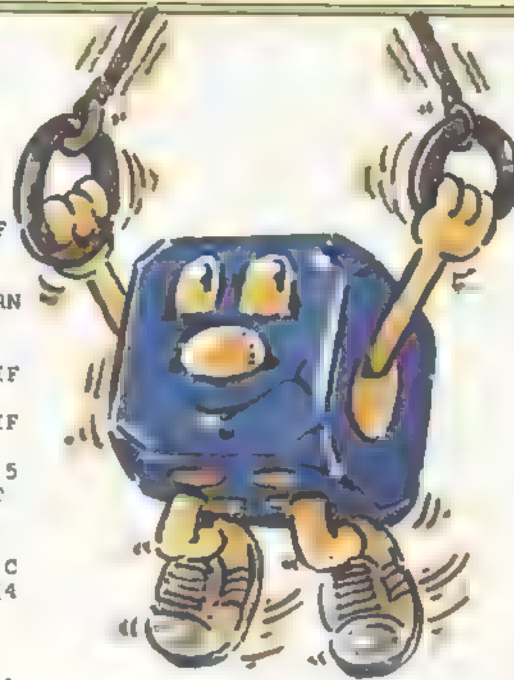


```
430 BORDER 0: PAPER 0: INK 6:
CLS
440 PLOT 4,4: DRAW 0,167: DRAW
124,0: DRAW 0,-167: DRAW -124,
0
450 PRINT  INK 5;AT 1,5;N$(N);
INK 4;AT 2,1;"** PLACAR **"
460 RESTORE 1280: FOR M=4 TO
17: READ A$: PRINT AT M,1;A$;
Q$( TO 11-LEN A$);: IF M<>16
THEN  PRINT ":"
470 NEXT M
480 GOSUB 530
490 GOSUB 560: GOSUB 530
500 PRINT  FLASH 1;AT 20,18;"Q
UALQUER TECLA";AT 21,18;"P/ CO
NTINUAR  "
510 LET A$=INKEY$: IF A$=""
THEN  GOTO 510
520 RETURN
530 FOR D=1 TO 12: IF P(N,D)=1
THEN  PRINT AT 3+D,13;"X"
540 IF O(N,D)<>0 THEN  PRINT
AT 3+D,13;O(N,D)
550 NEXT D: LET C=0: FOR D=1
TO 12: LET C=C+O(N,D): NEXT D:
PRINT AT 17,13;C: RETURN
560 PRINT AT 8,18;"TEMPO ";R;
AT 9,18;"SECAO ";I
570 PRINT AT 2,18;"RESULTADO=
": LET T=-1: GOSUB 1180
580 PRINT AT 5,18;"ESCOLHA O G
RUPO"
590 LET A=4
600 PRINT AT A,15;CHR$ 150
610 LET B$=INKEY$: IF B$=""
THEN  GOTO 610
620 IF B$=" " THEN  LET A=A-3:
GOTO 710
630 IF B$="K" THEN  GOTO 650
640 IF B$<>"M" THEN  GOTO 610
650 PRINT AT A,15;" "
660 IF B$="K" AND A=4 THEN
GOTO 600
670 IF B$="M" AND A=15 THEN
GOTO 600
680 IF B$="M" THEN  LET A=A+1
690 IF B$="K" THEN  LET A=A-1
700 SOUND .01,5: GOTO 600
710 PRINT AT A+3,15;" ": IF P(
N,A)<>0 THEN  GOTO 1240
720 IF A>6 THEN  GOTO 780
730 LET C=0
740 FOR D=1 TO 5: IF T(D)=A
THEN  LET C=C+1
750 NEXT D
760 LET O(N,A)=C*A
770 LET P(N,A)=1: RETURN
780 IF A=11 THEN  FOR D=1 TO 5
: LET O(N,11)=O(N,11)+T(D):
NEXT D: LET P(N,11)=1: RETURN
790 FOR D=1 TO 5: LET D(D)=0:
NEXT D: LET B=0: FOR E=1 TO 6:
LET C=0: FOR D=1 TO 5: IF T(D)
=E THEN  LET C=C+1
800 NEXT D: IF C<>0 THEN  LET
B=B+1
810 NEXT E
820 LET G=1: FOR F=1 TO 6:
GOSUB 1250: IF C<>0 THEN  LET
D(G)=F: LET G=G+1
830 NEXT F
840 LET P(N,A)=1: IF A=7 THEN
GOTO 950
850 IF A=8 THEN  GOTO 1010
```

```
860 IF A=9 THEN  GOTO 1050
870 IF A=10 THEN  GOTO 1120
890 IF A=12 THEN  GOTO 1160
950 IF B>2 THEN  GOTO 1190
960 IF B=1 THEN  GOSUB 1270:
 LET O(N,7)=C: RETURN
970 LET F=1
980 GOSUB 1250: LET F=F+1: IF
 C<>4 AND F<>7 THEN  GOTO 980
990 IF C<4 THEN  GOTO 1190
1000 LET O(N,7)=4*(F-1): RETURN

1010 IF B<>2 THEN  GOTO 1190
1020 LET F=D(1): GOSUB 1250: IF
 C=3 THEN  GOTO 1040
1030 LET F=D(2): GOSUB 1250: IF
 C<>3 THEN  GOTO 1190
1040 LET O(N,8)=0: FOR G=1 TO 5
: LET O(N,8)=O(N,8)+T(G): NEXT
G: RETURN
1050 IF B<>4 THEN  GOTO 1080
1060 GOSUB 1270: IF C<>18 AND C
<>10 AND C<>14 OR (C=14 AND D(4
)=6) THEN  GOTO 1190
1070 LET O(N,9)=15: RETURN
1080 IF B<>5 THEN  GOTO 1190
1090 GOSUB 1270: IF C=15 OR C=1
6 OR C=19 THEN  GOTO 1070
1100 IF C<>20 THEN  GOTO 1190
1110 GOTO 1070
1120 IF B<>5 THEN  GOTO 1190
1130 GOSUB 1270: IF C=15 OR C=2
0 THEN  GOTO 1150
1140 GOTO 1190
1150 LET O(N,10)=30: RETURN
1160 IF B<>1 THEN  GOTO 1190
1170 LET O(N,12)=50: RETURN
1190 SOUND .5,5: PRINT AT 20,18
;"ILEGAL !";AT 21,18;"ELIMINA ?
"
1200 LET A$=INKEY$: IF A$="" TH
EN  GOTO 1200
1210 IF A$="N" THEN  PRINT AT 2
0,18;"          ";AT 21,18;"
        ": LET P(N,A)=0: GOTO 590
1220 IF A$<>"S" THEN  GOTO 1200
1230 LET P(N,A)=1: RETURN
1240 SOUND .5,5: PRINT AT 20,18
;"GRUPO OCUPADO": FOR H=1 TO 30
0: NEXT H: PRINT AT 20,18;"
          ": GOTO 590
1250 LET C=0: FOR D=1 TO 5: IF
T(D)=F THEN  LET C=C+1
1260 NEXT D: RETURN
1270 LET C=0: FOR D=1 TO B: LET
 C=C+D(D): NEXT D: RETURN
1280 DATA "UNS","DOIS","TRES","
QUATROS","CINCOS","SEIS","IGUAI
S","CASA CHEIA","CURTO","LONGO"
,"MISTO","YATCH","            ",
"TOTAL"
```

```
350 CLS
360 W=6:Y=0:GOSUB 980:PRINT"***
*SCORE****"
370 PRINT"UNS.........:"X$"DOIS
........:"X$"TRES........:"X$"Q
UATROS.....:"X$"CINCOS......:"X
$"SEIS........:"
```



```
380 PRINT"4 OF A KIND.:"X$"FULL
 HOUESE..:"X$"SHORT RUN...:"X$"
LONG RUN....:"X$"SHORT RUN...:"
X$"LONG RUN....:"X$"CHOICE.....
.:"X$"YATCH.......:"X$
390 PRINT"TOTAL.......:";
400 GOSUB 460
410 GOSUB 490:GOSUB 460
420 PRINT @466,"qualquer tecla"
;:PRINT @498,"para continuar";
430 SOUND 60,1
440 A$=INKEY$:IF A$="" THEN 440
450 RETURN
460 FOR D=1 TO 12:IF P(N,D)=1 T
HEN PRINT @45+D*32,"X";
470 IF O(N,D)<>0 THEN PRINT @45
+D*32,O(N,D);
480 NEXT D:C=0:FOR D=1 TO 12:C=
C+O(N,D):NEXT:PRINT @493,C;:RET
URN
490 PRINT @53,"ROUND"R;:PRINT @
84,"SECAO"I;
500 PRINT @115,"GRUPO FINAL=";
510 FOR D=1 TO 5:FOR G=1 TO 4:P
RINT @111+G*32-(D>3)*118+4*D,D$
(T(D),G);:NEXT G,D
520 PRINT @403,"SELECT GROUP";
530 A=1
540 PRINT @49+32*A,CHR$(95);
550 B$=INKEY$:IF B$<>" " AND B$
<>"^" AND B$<>CHR$(10) THEN 550
560 IF B$=" " THEN 620
570 PRINT @49+32*A," ";
580 IF B$="^" AND A>1 THEN A=A-
1
590 IF B$=CHR$(10) AND A<12 THE
N A=A+1
600 SOUND 200,1
610 GOTO 540
620 PRINT @49+32*A," ";:IF P(N,
A)<>0 THEN 950
630 IF A>6 THEN 700
640 C=0
650 FOR D=1 TO 5:IF T(D)=A THEN
 C=C+1
660 NEXTD
670 O(N,A)=C*A
680 P(N,A)=1
690 RETURN
700 IF A=11 THEN FOR D=1 TO 5:O
(N,11)=O(N,11)+T(D):NEXT:P(N,11
)=1:RETURN
710 FOR D=1 TO 5:D(D)=0:NEXT:B=
0:FOR E=1 TO 6:C=0:FOR D=1 TO 5
:IF T(D)=E THEN C=C+1
720 NEXT D:IF C<>0 THEN B=B+1
730 NEXT E
740 G=1:FOR F=1 TO 6:GOSUB 1140
:IF C<>0 THEN D(G)=F:G=G+1
750 NEXT F
760 P(N,A)=1:A=A-6:ON A GOTO 77
0,810,830,870,960,890
770 IF B>2 THEN 900 ELSE IF B=1
 GOSUB 1160:O(N,7)=C*4:RETURN
780 F=1
790 GOSUB 1140:F=F+1:IF C<>4 AN
D F<>7 THEN 790 ELSE IF C<4 THE
N 900
800 O(N,7)=4*(F-1):RETURN
810 IF B<>2 THEN 900 ELSE F=D(1
):GOSUB 1140:IF C=3 THEN 820 EL
SE F=D(2):GOSUB 1140:IF C<>3 TH
EN 900
820 FOR D=1 TO 5:O(N,8)=O(N,8)+
T(D):NEXT:RETURN
830 IF B<>4 THEN 850 ELSE GOSUB
 1160:IF C<>18 AND C<>10 AND C<
>14 OR (C=14 AND D(4)=6) THEN 9
00
840 O(N,9)=15:RETURN
850 IF B<>5 THEN 900 ELSE GOSUB
 1160:IF C<>20 AND C<>15 AND C<
>16 AND C<>19 THEN 900
860 GOTO 840
870 IF B<>5 THEN 900 ELSE GOSUB
 1160:IF C<>20 AND C<>15 THEN 9
00
880 O(N,10)=30:RETURN
890 IF B<>1 THEN 900 ELSE O(N,1
2)=50:SOUND 5,8:FOR E=1 TO 700:
NEXT E:RETURN
900 SOUND 20,1:PRINT @432,"ileg
al. PERDE?";
910 A$=INKEY$:IF A$<>"S" AND A$
<>"N" THEN 910
920 PRINT @432,"
 ";
930 IF A$="N" THEN P(N,A+6)=0:G
OTO 530
940 P(N,A+6)=1:RETURN
950 SOUND 5,1:PRINT @433,"SECAO
 PREENCHIDA";:FOR E=1 TO 700:NE
XT:PRINT @433,"
";:GOTO 530
960 RETURN
1140 C=0:FOR D=1 TO 5:IF T(D)=F
 THEN C=C+1
1150 NEXT D:RETURN
1160 C=0:FOR D=1 TO B:C=C+D(D):
NEXT D:RETURN
```

IMPRESSÃO DO ESQUELETO

A primeira seção dessa rotina — que vai até a linha 520 no Spectrum e linha 450 nos outros micros — mostra na te-

la o esqueleto do placar (nomes dos grupos e do jogador) e chama duas outras rotinas para preencher os grupos e receber a última entrada.

A rotina constituída pelas três linhas seguintes preenche o esqueleto com o placar anterior e fornece a soma dos totais. Depois dela encontra-se uma rotina maior, incumbida de receber e analisar as entradas.

As primeiras linhas dessa rotina apresentam na tela informações sobre o número da partida e da jogada e, o que é mais importante, mostram a combinação final obtida. As linhas seguintes, além de identificar a tecla que está sendo pressionada, movem o cursor para cima e para baixo no esqueleto. Para a movimentação do cursor, o Spectrum usa as teclas M e K; o MSX e o TRS-Color, as setas, e o Apple e o TK-2000, as teclas Q e Z.

Quando se pressiona a barra de espaço, o programa interpreta que o grupo apontado pelo cursor foi escolhido. A posição do cursor é armazenada na variável **A**, onde **A** = 1 corresponde à posição do grupo dos UM, **A** = 2 equivale à posição do grupo dos DOIS e assim por diante até **A** = 11, que corresponde à posição do YATCH. O valor de **A** é usado pelo restante da rotina para calcular a pontuação do placar.

Uma outra variável importante é a variável **B**, que contém o número de dados diferentes entre si: **B** = 2 (dois tipos de valores), por exemplo, representaria uma FULL HOUSE e **B** = 5 (cinco tipos de valores), um LONGO.

Portanto, dependendo do valor de **A**, rotinas diferentes são chamadas para verificar se a combinação dos dados realmente corresponde ao grupo escolhido. Se corresponder, o placar é calculado e apresentado na tela. Caso contrário, a mensagem "ILEGAL" é exibida. O jogador decide, então — com um S ou um N —, se o grupo deve ser eliminado. O computador verifica também quais grupos já foram usados.

RESULTADO FINAL

Esta última seção simplesmente apresenta na tela o placar final, com os pontos dos jogadores em cada partida. O número da partida é exibido no topo do placar final e os nomes dos jogadores mais abaixo, à esquerda.

MSX

```
990 CLS·PRINT">>>>>>>>>>    PLAC
AR FINAL   <<<<<<<<<<"
1000 LOCATE0,4·FORD=1TONP:PRINT
N$(D):NEXT
1020 FORD=1TONP
1030 C=0:FORE=1TO12:C=C+O(D,E)·
NEXT·S(D,R)=C·NEXTD
1040 FORD=1TO5:LOCATE3+D*4,2:PR
INTD;·NEXT
1050 PRINT" TOTAL"
1060 FORD=1TONP
1070 FORE=1TOR
1080 LOCATE3+E*4,3+D:PRINTS(D,E
);:NEXTE,D
1090 FORD=1TONP·C=0·FORE=1TOR:C
=C+S(D,E):NEXTE·LOCATE26,3+D·PR
INTC;·NEXTD
1100 LOCATE2,19:PRINT"< qualque
r tecla para outra rodada >"
1110 A$=INKEY$·IFA$=""THEN1110
1120 FORE=1TONP:FORD=1TO12:O(E,
D)=0:P(E,D)=0:NEXTD,E
1130 RETURN
```

Apple / TK-2000

```
990  HOME · INVERSE · PRINT ">>
>>>>>>>>  PLACAR FINAL  <<<<<<
<<<<<": NORMAL
1000  VTAB (5)  FOR D = 1 TO NP
· PRINT  LEFT$ (N$(D),10)· NEXT

1020  FOR D = 1 TO NP
1030 C = 0: FOR E = 1 TO 12:C =
 C + O(D,E): NEXT :S(D,R) = C:
NEXT D
1040  FOR D = 1 TO 5: HTAB (9 +
 D * 4): VTAB (3): PRINT D;  NE
XT
1050  PRINT "   TOTAL"
1060  FOR D = 1 TO NP
1070  FOR E = 1 TO R
1080  HTAB (9 + E * 4): VTAB (4
 + D): PRINT S(D,E);: NEXT E,D
1090  FOR D = 1 TO NP:C = 0: FO
R E = 1 TO R:C = C + S(D,E): NE
XT E· HTAB (33): VTAB (4 + D)·
PRINT C;· NEXT D
1100  HTAB (2)· VTAB (22)· PRIN
T "< QUALQUER TECLA PARA OUTRA
RODADA >";
1110  GET A$: IF A$ = "" THEN 1
110
1120  FOR E = 1 TO NP: FOR D =
1 TO 12:O(E,D) = 0:P(E,D) = 0:
NEXT D,E
1130  RETURN
```

Spectrum

```
1290 BORDER 7· PAPER 7. INK 3·
CLS
1300 PRINT "**********PLACAR FI
NAL**********"
1310 PRINT AT 7,0: FOR D=1 TO N
P· PRINT N$(D);" :"'': NEXT D
1320 PRINT  INK 1;AT 3,0;"JOGAD
OR"; INK 2;AT 3,13;"T E M P O"
1330 PRINT AT 5,12;: FOR D=1 TO
 5· PRINT  INK 2;TAB (4+D*4),D;
: NEXT D: PRINT  INK 0;"   TOTE
"
1340 FOR D=1 TO NP
1360 LET C=0  FOR E=1 TO 12: LE
T C=C+O(D,E): NEXT E· LET S(D,R
)=C: NEXT D
1370 FOR D=1 TO NP: FOR E=1 TO
■
1380 PRINT  INK 4;AT 6+D*2,4+E*
4;S(D,E): NEXT E: NEXT D
1390 FOR D=1 TO NP: LET C=0: FO
R E=1 TO R: LET C=C+S(D,E): NEX
T E: PRINT AT 6+D*2,28;C: NEXT
D
1400 PRINT #1; INVERSE 1; INK 0
;" QUALQUER TECLA PARA CONTINUA
R "
1410 LET A$=INKEY$· IF A$="" TH
EN  GOTO 1410
1420 FOR E=1 TO NP: FOR D=1 TO
12· LET O(E,D)=0· LET P(E,D)=0:
 NEXT D: NEXT E
1430 RETURN
```

TRS-Color

```
990 CLS:PRINT"*****TABELA DE PO
NTOS*****"
1000 PRINT @128:FOR D=1 TO NP:P
RINT N$(D):NEXT
1010 PRINT @75,"R O U N D"
1020 FOR D=1 TO NP
1030 C=0:FOR E=1 TO 12:C=C+O(D,
E):NEXT:S(D,R)=C:NEXT D
1040 FOR D=1 TO 5:PRINT @98+D*4
,D;:NEXT
1050 PRINT" TOTAL"
1060 FOR D=1 TO NP
1070 FOR E=1 TO R
1080 PRINT @129+D*32+E*4,S(D,E)
;:NEXT E,D
1090 FOR D=1 TO NP:C=0 FOR E=1
TO R·C=C+S(D,E):NEXT E:PRINT @1
53+D*32,C;:NEXT D
1100 PRINT @448,"  QUALQUER TEC
LA PARA PROXIMA     JOGADA.";
1110 A$=INKEY$:IF A$="" THEN 11
10
1120 FOR E=1 TO NP:FOR D=1 TO 1
2:O(E,D)=0:P(E,D)=0:NEXT D,E
1130 RETURN
```

AVALANCHE: AS COBRAS VIVEM!

- NÍVEL DO JOGO
- O ATRASO DAS COBRAS
- MOVIMENTO DAS LÍNGUAS
- A COBRA ESTÁ SUBMERSA?
- SINCRONISMO

As cobras ainda não constituem ameaça: podem ser vistas dentro dos buracos, mas estão sempre quietas. Agora vamos dar-lhes movimento, complicando um pouco mais a vida do pobre Willie.

Nossa aventura está bem movimentada: Willie anda e salta, as pedras rolam, os pássaros voam. As cobras, porém, permanecem totalmente imóveis em seus buracos. Está na hora de lhes dar um pouco de vida.



A rotina a seguir atiça as cobras, fazendo-as sair de sua inatividade.

```
10 REM org 59823
20 REM snk ld a,(57344)
30 REM cp 2
40 REM jr nc,sko
50 REM ret
60 REM sko ld ix,57350
70 REM ld hl,425
80 REM call skm
90 REM ld hl,369
100 REM call skm
110 REM ld hl,282
120 REM call skm
130 REM ret
```

```
140 REM skm push hl
150 REM ld de,(57354)
160 REM sbc hl,de
170 REM pop hl
180 REM jr c,sns
190 REM ret
200 REM sns ld a,(ix+0)
210 REM inc a
220 REM res 4,a
230 REM ld (ix+0),a
240 REM inc ix
250 REM cp 7
260 REM jr nc,sco
270 REM ret
280 REM sco ld bc,57224
290 REM ld d,43
300 REM cp 15
310 REM jr nz,ste
320 REM ld bc,15616
330 REM ld d,45
340 REM ste ld a,d
350 REM call 58217
360 REM ret
```

A primeira tarefa da rotina consiste em verificar se as cobras devem entrar em ação. Para isso, o valor do nível do jogo, armazenado em 57344, é carregado no acumulador e comparado com 2. Essa comparação é feita por meio de uma subtração cujo resultado não armazenamos. Se compararmos 2 com um número menor, teremos um resto. A baliza *carry* será então ajustada com 1 e o processador retornará, porque as co bras não são necessárias.

Se a baliza carry não foi ajustada, a instrução **jr nc,sko** pula a instrução **ret**. Você estará no terceiro ou quarto nível do jogo (valor 2 ou 3) e as cobras devem ser atiçadas.

AS COBRAS AMEAÇAM

A posição de memória 57350 carrega o primeiro dos atrasos das cobras, que indicam quando elas devem mexer a língua. IX é carregado com esse endereço, e HL, com 425, a posição da primeira língua de cobra. A rotina **skm**, que executa o movimento, é chamada.

A posição da língua da próxima cobra é carregada em HL, e **skm** volta a ser chamada. Em seguida, a posição da língua da terceira cobra é carregada em HL e **skm** é chamada mais uma vez.

Quando todas as línguas estiverem se mexendo, a rotina retorna.

SE A MARÉ SOBE...

Quando a rotina **skm** é chamada, as posições das línguas são colocadas na pilha para armazenamento temporário. A posição do mar em 57354 é colocada no par DE. Neste ponto do programa, precisamos verificar a altura da maré. Se a cobra estiver submersa, a parte da rotina que movimenta sua língua pode ser pulada.

A seguir, a posição do mar no par DE é subtraída da posição da língua em HL. Se não houver resto, a posição do mar se encontra acima da posição da língua da cobra na tela. Nesse caso, a instrução **jr c,sns** não tem efeito e o processador retorna. Mas, se houver resto, ou seja, se a posição do mar estiver abaixo da posição da língua na tela, a instrução **jr c,sns** faz com que o processador salte a instrução **ret** e continue a rotina.

ESTÁ NA HORA DE MEXER?

O atraso da cobra é carregado da posição apontada por IX para o acumulador, onde é incrementado. O bit 4 é apagado para evitar que o atraso se torne maior que 15. O resultado dessa operação é armazenado de volta na posição indicada por IX. Esse apontador é então incrementado para indicar o atraso da cobra seguinte. Seu valor é comparado com o conteúdo do acumulador pela instrução **cp 7**. Se o atraso dessa cobra for maior ou igual a 7, **jr nc,sno** manda o processador para a rotina que imprime a língua. Se o valor for menor que 7, a baliza carry é ajustada com 1, a instrução não tem efeito e o processador retorna.

Cada vez que essa rotina é chamada, um atraso diferente da cobra é encontrado nos endereços 57350, 57351 e 57352, pois o apontador IX é incrementado entre cada chamada que se faz à rotina **skm**. A língua da cobra fica para fora oito ciclos, e não aparece nos oito ciclos seguintes.

IMPRESSÃO

Chegamos, finalmente, à rotina encarregada de imprimir ou apagar a língua da cobra na tela. O par de registros BC é carregado com 57224, os dados para a língua da cobra. D é ajustado com 43, a cor da figura. O atraso da cobra em A é comparado com 15.

Se o atraso não tiver chegado a 15, a instrução **jr nz** leva o processador pa-



ra **ste**. Se for igual a esse valor, você precisará apagar a língua, para que ela não apareça nos próximos oito ciclos. BC é carregado com 15616, o endereço dos dados de um espaço vazio, e o registro D é carregado com 45, o código da cor do céu.

Qualquer que seja o valor do atraso da cobra (neste ponto da rotina, ele é sempre maior do que sete), o processador encontra a instrução **ld a,d**. Depois que ela carrega a cor apropriada em A, a rotina **print**, em 58217, é chamada. Com isso, imprimimos a língua — ou apagamos a que existia — para fora da boca de uma das cobras. A seguir, o processador retorna.

T 

Esta rotina movimenta as cobras, acrescentando ação ao jogo.

```
10   ORG 20856
20   SNK LDA 18238
30   CMPA #2
40   BHS SKO
50   RTS
60   SKO LDY #18255
70   LDX #5095
80   JSR SKM
90   LDX #4591
100   JSR SKM
110   LDX #3833
120   JSR SKM
130   RTS
140   SKM PSHS X
```

MICRO DICAS

PROGRAMAS LONGOS: MELHORE A VELOCIDADE DE MONTAGEM

Como o leitor deve ter notado, apresentamos o programa *Avalanche* em pequenos segmentos funcionais, que podem ser testados separadamente. Essa técnica de construção de programas, chamada de *desenvolvimento modular*, é muito útil para qualquer tipo de programa, pois agiliza o processo de montagem e facilita enormemente os testes de execução. Para a programação complexa em linguagem de máquina, o uso dessa técnica é essencial, devido à dificuldade de se documentar internamente — isto é, na própria listagem do programa — o código em Assembler.

Os profissionais que criam os sofisticados videogames que vemos em fliperamas, por exemplo, dominam perfeitamente a técnica de modularização. Na realidade, eles dispõem de programas Assembler poderosíssimos, capazes de representar uma biblioteca de rotinas e ferramentas de programação que facilitam muito a implementação dos recursos normalmente usados em videogames.

Infelizmente, os Assembler que *INPUT* apresentou para os diferentes micros são muito lentos. Se você quiser melhorar a velocidade de montagem, divida um programa mais longo em segmentos menores e compile-os separadamente. Não se esqueça, porém, de calcular o endereço de origem para cada segmento e incluí-lo em um comando org ou equivalente, no começo de cada segmento.

```
150  LDX 18247
160  LEAX 31,X
170  CMPX ,S
180  PULS X
190  BHI SNS
200  RTS
210  SNS LDA ,Y
220  INCA
230  ANDA #$F
240  STA ,Y+
250  CMPA #7
260  BHS SCO
270  RTS
280  SCO LDU #18062
290  CMPA #15
300  BNE STE
310  LDU #1536
320  STE LEAX -256,X
330  JSR CHARPR
340  RTS
350  CHARPR EQU 19402
```

A primeira tarefa da rotina é verificar se as cobras devem entrar em ação. Para isso, o valor do nível do jogo, armazenado em 18238, é carregado no acumulador e comparado com 2.

Se você está no nível três ou quatro — ou seja, se o valor é 2 ou 3 —, as cobras serão necessárias. Nesse caso, a instrução **BHS** faz o processador pular o **RTS**, seguindo a rotina.

AS COBRAS AMEAÇAM

Y é carregado com o endereço do atraso da primeira cobra. Esta é a variável incumbida de interromper ou começar o seu movimento. Os atrasos da segunda e da terceira cobras estão logo em seguida na memória.

X é carregado com 5095, a posição na tela do primeiro buraco, e o processador vai para a rotina **SKM**.

Essa rotina atiça a primeira cobra e incrementa o conteúdo de Y antes de retornar. Assim, quando o processador salta para fazer com que a segunda e a terceira cobras se movimentem, apenas as posições do segundo e do terceiro buracos precisam ser carregadas em X. Y é incrementado automaticamente, apontando os atrasos das três cobras. Quando tiver passado por todas elas, o processador retorna.

MARÉ ALTA

A primeira coisa que o processador faz ao entrar na rotina **SKM** é colocar na pilha, para armazenamento temporário, a posição da cobra na tela.

O registrador X é carregado com o conteúdo de 18247, a posição do mar. Esse valor é somado com 31, para que o apontador de tela se mova para o canto direito da tela. X é então comparado com a posição do buraco da cobra, que está na pilha de máquina, através da instrução **CMPX ,S**. A posição da cobra é recuperada da pilha, voltando para o registrador X.

Como isso não afeta nenhuma das balizas, a instrução BHI ainda se refere à operação **CMPX ,S**. Assim, se o mar está acima do buraco — ou seja, se a posição do mar na tela não é maior do que a posição do buraco que estava na pilha —, o salto não ocorre, e o processador retorna.

Caso o mar não tenha encoberto o buraco — isto é, se a posição do mar na tela é maior do que a posição do buraco —, o processador pula a instrução **RTS**, e dá continuidade à rotina que atiça as cobras.

MEXER OU NÃO MEXER

A é carregado com o conteúdo do atraso da cobra que está apontado no registrador Y. Esse valor é então incrementado e a operação AND é feita com $F, para ajustá-lo com 0 toda vez que ultrapassar o valor 15.

O resultado é armazenado de volta no endereço apontado por Y. Esse registrador é então incrementado para apontar o atraso da próxima cobra.

O atraso da cobra que ainda está em A é comparado com 7. Se for maior ou igual, o processador vai para a rotina de impressão da língua; caso contrário, o processador retorna.

O BOTE

Ao entrar na rotina **SCO**, o processador carrega o apontador da pilha do usuário com 18062, o endereço dos dados para a língua da cobra.

A — que ainda contém o atraso da cobra que foi incrementado — é comparado com 15. Se ele não foi incrementado até esse valor, o processador pula para as instruções de impressão. Mas, se já chegou a 15, U é carregado com 1536, o endereço da parte de céu no canto superior esquerdo da tela.

Seguindo adiante, X é subtraído de 256, fazendo o apontador de tela se mover um caractere para cima do buraco da cobra. Esta é a posição da língua.

Se o atraso da cobra estiver entre 7 e 14, a rotina **CHARPR** imprimirá a língua nessa posição ao ser chamada. Mas, se o atraso tiver atingido o valor 15, a língua será apagada.

A língua permanecerá invisível, isto é, não será impressa, até que o atraso chegue novamente a 7.



A rotina apresentada a seguir atiça as cobras que até agora permaneciam quietas em seus buracos.

```
10   org 55564
20   ld a,(-5228)
30   cp 2
40   jr nc,sk
50   ret
60   sk ld bc,-5198
70   ld hl,425
80   call sm
90   ld hl,369
100   call sm
110   ld hl,282
120   call sm
130   ret
140   sm push hl
150   ld de,(-5212)
```



```
160    sbc hl,de
170    pop hl
180    jr c,sn
190    ret
200    sn ld a,(bc)
210    inc a
220    res 4,a
230    ld (bc),a
240    inc bc
250    cp 7
260    jr nc,sc
270    ret
280    sc ld de,(62407)
290    add hl,de
300    cp 15
310    ld b,36
320    jr nz,st
330    ld b,255
340    st ld a,b
350    call 77
360    ret
370    end
```

A primeira tarefa da rotina consiste em verificar se as cobras devem entrar em ação no nível atual do jogo. Para isso, o valor do nível, −5228, é carregado no acumulador e comparado com 2. Se você está no terceiro ou no quarto nível — ou seja, se o valor é 2 ou 3 — as cobras estão nos buracos e devemos atiçá-las.

A comparação é feita por meio de uma subtração cujo resultado não armazenamos. Se compararmos 2 com um número menor, teremos um resto e a baliza carry passará a conter 1. Nesse caso, o processador retornará, porque as cobras não são necessárias.

Caso a baliza carry não tenha sido afetada, a instrução **jr nc,sk** pula a instrução **ret**, porque as cobras estão presentes neste nível.

AS COBRAS AMEAÇAM

O endereço de memória −5198 contém o primeiro dos chamados atrasos das cobras, que indicam quando elas devem mostrar sua língua. O par BC é carregado com esse endereço, e HL, com 425, a posição da língua da primeira cobra. A rotina **sm**, encarregada de promover o movimento, é chamada.

A posição da língua da próxima cobra é carregada em HL, e **sm** volta a ser chamada. Em seguida, a posição da língua da terceira cobra é carregada em HL, e **sm** é chamada mais uma vez.

Depois de movimentar todas as línguas, a rotina retorna.

MARÉ ALTA

Quando a rotina **sm** é chamada, a posição da língua que está em HL é colocada na pilha para armazenamento temporário. A posição do mar em −5212 é colocada no par DE. Neste ponto do programa, precisamos verificar se a cobra se afogou com a subida da maré. Em caso afirmativo, ela não precisa mexer a língua — assim, a parte da rotina que faz isso pode ser pulada.

Para essa verificação, a posição do mar no par DE é subtraída da posição da língua em HL. Se não houver resto — ou seja, se a posição do mar estiver acima da posição da língua da cobra na tela —, a instrução **jr c,sn** não tem efeito e o processador retorna. Mas, se houver resto, a posição do mar está abaixo da posição da língua na tela. A instrução **jr c,sn** faz, então, o processador saltar a instrução **ret** e continuar a rotina.

ESTÁ NA HORA DE MEXER?

O atraso da cobra é carregado da posição apontada por BC para o acumulador, onde é incrementado. O bit 4 também é incrementado para evitar que o atraso se torne maior que 15. O resultado dessa operação é armazenado de volta na posição apontada por BC.

O par de registros BC é incrementado para apontar o atraso da cobra seguinte. A instrução **cp 7** compara esse valor com o conteúdo do acumulador, e a instrução **jr nc,sc** manda o processador para a rotina que imprime a língua, se o atraso da cobra for maior ou igual a 7. Caso o valor seja inferior a 7, a baliza é ajustada com 1, a instrução de desvio não tem efeito e o processador retorna.

Ao ser chamada, essa rotina sempre encontra um atraso diferente nos endereços −5198, −5199 e −5220, já que o apontador BC é incrementado entre cada acesso a **sm**. A língua da cobra fica para fora oito ciclos, não aparecendo nos oito seguintes.

IMPRESSÃO

Chegamos, finalmente, à rotina que imprime ou apaga a língua da cobra na tela. O endereço inicial da Tabela de Nomes da VRAM é carregado em DE e somado a HL, que contém a posição da língua. Esse par de registros passa então a conter o endereço correspondente à posição da língua na TN.

O atraso da cobra que ainda está no acumulador é comparado com 15. O registro B é carregado com 36, o código do padrão da língua. Se o valor do atraso não for 15, a instrução **jr nz,st** leva o processador para **st**, pulando a instrução **ld b,255**. Caso contrário, a língua deve ser apagada para que não apareça nos próximos oito ciclos; o desvio não ocorre e 255, o código do padrão de céu, é colocado em B.

Neste ponto do programa, o valor do atraso da cobra será sempre superior a 7. O processador encontra a instrução **ld a,b**, que transfere o código do padrão apropriado de B para A. A rotina 77 da ROM, que coloca o código que está em A na posição da TN apontada por HL, é chamada. Com isso, imprimimos a língua — ou apagamos a que existia — para fora da boca da cobra. A seguir, o processador retorna ao laço principal do jogo.

O SISTEMA OPERACIONAL

Sem o sistema operacional, seria difícil trabalhar com o computador. Veja de que modo funciona esse importante componente do seu micro e aprenda a fazer dele um aliado.

Todos os computadores trabalham e entendem apenas uma linguagem: o código de máquina, que é representado internamente por pequenas variações binárias de voltagem. Ele pode ser expresso por um conjunto de letras e números, o que facilita sua compreensão, ou por números binários, menos familiares ao usuário. De uma forma ou de outra, essa linguagem é considerada pouco acessível por muitas pessoas. O que torna mais simples a comunicação com o operador é o Sistema Operacional do computador (SO), um conjunto de rotinas em código de máquina que controlam todas as funções da máquina. Seja qual for a linguagem empregada, BASIC ou outra qualquer, essas rotinas serão sempre utilizadas para controlar as diversas funções do computador.

Conhecer o funcionamento do SO não é essencial para o usuário, porém amplia bastante suas alternativas de programação. O uso de funções internas do SO possibilita a obtenção de melhores resultados não só quanto à velocidade, mas, também, quanto à qualidade geral do programa.

Dentro dos limites impostos por cada máquina, este artigo procura mostrar



algumas aplicações úteis para as rotinas e variáveis próprias de cada sistema operacional.

O QUE É SISTEMA OPERACIONAL

O SO é apenas um tipo sofisticado de programa — escrito em código de máquina — que permite ao microprocessador existente dentro da máquina responder aos comandos básicos de operação. Ele controla a comunicação da máquina com o mundo exterior — o teclado, a tela, o alto-falante e outras portas de entrada e saída.



O SO tem ainda a função de assegurar que a memória seja utilizada de forma eficiente pelo usuário e pelo interpretador BASIC. Assim, quando ligamos o computador, ele ativa várias de suas rotinas da ROM para acertar os valores iniciais de variáveis e apontadores e, em seguida, avisa que está pronto a aceitar comandos.

Em todos os microcomputadores abordados em *INPUT*, o SO é responsável pela interpretação dos comandos em BASIC, contudo ele pode também lidar com outros tipos de linguagem.

Na maior parte do tempo, o trabalho do SO consiste em verificar e modificar os valores de determinadas variáveis e apontadores especiais. É importante, por exemplo, que o SO conheça o endereço exato tanto do início como do fim de um programa em BASIC. Se o programa for modificado, seu tamanho também mudará, e esses valores precisarão ser atualizados.

O mesmo se aplica ao processo de alocação e manutenção de espaço para as variáveis, particularmente as variáveis indexadas, ou matrizes. Quando dimensionamos uma variável desse tipo, um espaço deve ser reservado — e, quando a variável não for mais necessária, ele deve ser eliminado.

É essencial que o SO utilize racionalmente a memória, pois, caso contrário, um programa longo ou algumas variáveis mais compridas logo esgotarão a quantidade de espaço disponível. Essa administração da memória é chamada de *housekeeping* — que poderíamos traduzir livremente por "cuidar da casa". Uma boa "economia doméstica" é indispensável para que o computador funcione eficientemente.

O SO contém ainda uma série de rotinas de software montadas de modo que tanto o usuário quanto o próprio SO possam utilizá-las com facilidade. A maioria delas não é de interesse para o programador em BASIC, pois vários comandos dessa linguagem se encarregam da execução das mesmas tarefas, chamando as rotinas apropriadas. Um bom exemplo é o comando **INPUT**, que usa uma série de sub-rotinas do SO, tais como seleção e leitura da porta de entrada, varredura do teclado, uso e transferência de buffer etc.

Outro exemplo interessante é **CLEAR**. Esse comando força o SO a executar todas as rotinas necessárias à liberação da memória RAM disponível para o usuário, destruindo todas as variáveis previamente existentes.

TRABALHO SOB ENCOMENDA

Sempre que digitamos um comando, o SO seleciona e coloca em ação uma das rotinas de seu vasto acervo. Suponhamos que você tecle a letra A: o SO deverá instruir o processador a imprimir o caractere na tela, cabendo-lhe também detectar a tecla que foi pressionada. A forma como ele faz isso varia de computador para computador. Em certas máquinas, o SO "varre" periodicamente o teclado para verificar se alguma tecla foi pressionada. Em outras, seu funcionamento normal é interrompido sempre que se pressiona uma tecla, a qual é identificada, em seguida, por meio de uma varredura.

Para responder adequadamente à pressão da tecla, o SO aciona a sub-rotina que manda o caractere correspondente à tela. Sua execução, assim como a de outras rotinas em código de máquina, é muito rápida. Se pressionarmos a tecla **<ENTER>** ou a tecla **<RETURN>**, o SO ativará uma rotina que promove a mudança de linha na tela (*new line*).

O INTERPRETADOR BASIC

Quando executamos um programa codificado em uma linguagem como o BASIC, as instruções que estão na memória são interpretadas, ou seja, traduzidas para os códigos que o computador pode entender. Esses códigos modificam os registradores de modo que as rotinas do SO possam ser chamadas adequadamente. Assim, as mesmas rotinas que possibilitam a execução de comandos

entrados pelo teclado permitem ao SO a execução de um programa BASIC.

Mas por que o BASIC é tão lento, já que utiliza rotinas em código, que são rápidas? Acontece que a linguagem deve ser interpretada a partir de palavras-chave — como **PRINT** — e de outros símbolos, antes que essas rotinas internas possam ser executadas. Como a tradução do BASIC é demorada, programas que exigem rapidez, como os de videogame, por exemplo, são escritos diretamente em linguagem de máquina.

O tempo que se leva para escrever um programa em codigo de máquina é, em média, dez vezes maior do que o gasto em um programa BASIC. A velocidade de execução do mesmo, contudo, chega a ser cinqüenta vezes maior. Assim, a não ser que a rapidez de execução seja muito importante, a maioria das pessoas prefere utilizar o BASIC, dada a facilidade de programação.

ACESSO DIRETO AO SO

A possibilidade de ter acesso direto ao SO, sem passar pelo interpretador, seria a solução ideal para quem prefere as facilidades do BASIC, mas, por outro lado, não quer abrir mão da velocidade. Nem todas as máquinas, porém, oferecem essa alternativa.

No TRS-80 e no TRS-Color, por exemplo, o SO faz parte do interpretador de dezesseis Kbytes — o Microsoft BASIC. No MSX, além de um conjunto de rotinas básicas, chamado BIOS (*BASIC Input/Output System*), o SO inclui também o interpretador. Os micros da linha Sinclair, como o ZX-81 e o Spectrum, são ainda mais restritivos, uma vez que não permitem acesso aos registradores internos diretamente do BASIC (o que é essencial para que se possa selecionar as rotinas existentes no SO).

Seja qual for o microcomputador e o tipo de acesso que oferece ao SO, sempre é possível contornar diversos problemas — inclusive a diminuição do tamanho das rotinas em código — por meio de alguns truques.

VARIÁVEIS DO SISTEMA

Os usuários do Spectrum podem utilizar as rotinas do SO através do comando **USR**, que também serve para executar rotinas em código de máquina colocadas temporariamente na RAM. Tente usar, por exemplo, o comando direto **RANDOM USR 0**: ele provoca um *reset* geral na máquina (reinicialização). Para programas em BASIC, há a alternativa de se modificar determinadas variáveis por meio de comandos **POKE**.

POKE 23561, seguido de um número entre 1 e 255, altera o lapso de tempo que antecede a ativação da auto-repetição das teclas. O valor normal é dado por um **POKE 23561,35**. De modo análogo, **POKE 23562,5**, que define o valor normal do período que decorre entre duas auto-repetições sucessivas, também pode ser modificado. Tal recurso é útil em jogos, ou qualquer outro tipo de programa que exija respostas rápidas do usuário, via teclado.

As posições 23606 e 23607 contêm o endereço dos padrões de pontos dos caracteres. Se usarmos **POKE 23606,8** (byte menos significativo), o apontador será movido um caractere para cima, na tabela. Assim, qualquer letra que digitarmos aparecerá na tela como o caractere seguinte do código ASCII. Tente digitar esse **POKE** seguido de 1, 2, 3, 4. Note que este é um método simples de "criptografar" listagens em BASIC, desencorajando curiosos que tentem lê-las. Se, em vez do byte menos significativo você recorrer ao mais significativo — **POKE 23607,0** —, o apontador será dirigido para o começo da memória ROM, tornando os caracteres absolutamente ininteligíveis.

A posição de memória 23658 possibilita alterar o estado da tecla **<CAPS LOCK>** durante a execução de um programa. **POKE 23658,0** muda os caracteres de maiúsculos para minúsculos; **POKE 23658,8** admite só maiúsculos.

O comando **PLOT** permite especificar uma posição absoluta da tela, enquanto **DRAW** se refere de forma relativa à posição corrente do cursor gráfico. Se você precisar de um **DRAW** absoluto, poderá consegui-lo através das posições de memória 23677 e 23678. Para verificar seu efeito, digite **PLOT 128,85.** Esse comando colocará um ponto no centro da tela. Suponhamos que você queira traçar uma linha que vá deste ponto até o canto superior direito da tela, cuja posição absoluta é (255,175). **DRAW 255,175** indicaria um ponto fora da tela, mas **DRAW 255-PEEK 23677,175-PEEK 23678** lhe dará o resultado desejado. Ao subtrair **PEEK 23677** da coordenada X e **PEEK 23678** da coordenada Y, estamos, na verdade, efetuando uma operação para obter as coordenadas relativas do comando **DRAW**.

Um outro exemplo de utilização das rotinas do SO por meio de um comando BASIC é dado pela simulação de um relógio. No Spectrum, o contador de linhas de vídeo ocupa as posições 23672, 23673, 23674. Se colocarmos 0 nelas, usando o comando **POKE**, o contador será zerado. Depois disso, sua atualização será feita automaticamente através de interrupções. Aqui está um programa simples que emprega o contador para improvisar um relógio:

```
 10 POKE 23674,0: POKE 23673,0
 : POKE 23672,0
 20 BORDER 0: PAPER 0: INK 6:
 CLS
 30 DEF FN t()=INT ((65536*
 PEEK 23674+256*PEEK 23673+
 PEEK 23672)/50)
 40 LET t=FN t()
 50 LET h=INT (t/(60*60))
 60 LET m=INT (t/60)
 70 LET s=t-((h*60)*60)-(m*60)
 80 LET t$="["+STR$ h+":"+STR$
 m+":"+STR$ s+"]"
 90 PRINT AT 1,15-((LEN t$)/2)
 :" ";t$;" "
100 GOTO 100
```

A linha 10 zera o contador de linhas de vídeo, enquanto a linha 30 define a função t, que o lê. A linha 40 armazena o valor encontrado em t. A partir desse valor, as horas, minutos e segundos são calculados.

ENTRADA E SAÍDA

Os microcomputadores pertencentes à linha MSX possuem cerca de 32 Kbytes de memória ROM dedicados ao SO. Os primeiros 16385 bytes são preenchidos com rotinas de entrada e saída. O interpretador BASIC começa em 16385 e termina em 32769. Existem ainda 3202 bytes dedicados às variáveis do sistema, todos eles situados no topo da memória — 62333 a 65535.

Um SO com essa extensão possui quase todas as rotinas que o usuário pode desejar. Muitas dessas rotinas têm sido aproveitadas no videogame *Avalanche*. Contudo, como a maioria delas emprega valores armazenados em registradores internos do Z-80 — o microprocessador do MSX —, poucas podem ser utilizadas diretamente por programas em BASIC. A solução consiste em escrever pequenas rotinas em código que modificam os valores dos registradores, de modo que o BASIC possa chamar a rotina desejada do SO. Um exemplo desse procedimento foi dado no artigo publicado à página 1141 de *INPUT*.

De fato, é uma pena que o BASIC não possa modificar os registradores diretamente, sobretudo porque o SO do MSX dispõe de certos artifícios projetados para facilitar a utilização de suas rotinas pelo usuário. Algumas porções

da ROM e da RAM são preenchidas com vetores que apontam para determinadas rotinas do SO, numa tentativa de simplificar seu uso.

As rotinas de entrada e saída — comunicação com a tela, com o gerador de som, com o teclado e o cassete, entre outros periféricos — estão espalhadas pelos primeiros 16K da ROM. Existe, porém, uma pequena região — que vai do endereço 59 ao 348 — que contém vetores que apontam para as principais rotinas do BIOS. Assim, podemos ter acesso a essas rotinas de duas maneiras: chamando seu endereço verdadeiro na memória ROM ou usando seu vetor na tabela. As rotinas do interpretador BASIC também são apontadas por vetores de uma tabela localizada no topo da memória — 64922 a 65535.

As rotinas mais interessantes necessitam de parâmetros colocados nos registradores internos. Como muitas delas já foram apresentadas em outros artigos publicados em *INPUT*, faremos uma pequena lista de rotinas (endereços da tabela de vetores) e endereços de variáveis úteis

As rotinas podem ser usadas por meio dos comandos **DEFUSR** e **USR()**, e as variáveis, por meio de **PEEK** e **POKE**.

GRÁFICOS

Eis algumas rotinas de interesse na utilização da tela:

65 - Desabilita a geração de imagens na tela. Pode ser útil quando se pretende que o usuário não acompanhe a elaboração de um desenho complicado, vendo-o só depois de completo.
68 - Reabilita a geração de imagens. Faz com que o processo de desenho, que a rotina anterior oculta, apareça instantaneamente na tela.
98 - Muda as cores da tela de acordo com o valor das variáveis do sistema.
108 - Equivale a **SCREEN 0**. Os endereços das tabelas e as cores podem ser modificados através das variáveis do sistema.
111 - **SCREEN 1**.
114 - **SCREEN 2**.
117 - **SCREEN 3**.
192 - Emite um sinal sonoro.
195 - **CLS**.
207 - Mostra as teclas de função na parte inferior da tela.
204 - Apaga as teclas de função da tela.

São variáveis de especial interesse para a tela:

62387 a 62426 - Cada par de bytes corresponde ao endereço de uma **BASE**. Por exemplo, 62407 e 62408 contêm o endereço de **BASE(10)**, tabela de nomes da tela gráfica.
62431 a 62438 - Conteúdo dos registradores do VDP (*Video Display Processor*).
62441 - Cor de frente.
62442 - Cor de fundo.
62443 - Cor da borda.
64695 e 64696 - Coordenada X.
64697 e 64698 - Coordenada Y.

TECLADO E CASSETE

Apresentamos a seguir algumas rotinas de comunicação com o teclado e o gravador cassete. A maioria dessas rotinas precisa de determinados parâmetros nos registradores.

159 - Aguarda que uma tecla seja pressionada pelo usuário e devolve seu código ASCII ao registrador A. Essa rotina pode substituir a linha em BASIC que geralmente aparece depois que a mensagem "APERTE QUALQUER TECLA" é impressa:

```
100 IF INKEY$=" " THEN 100
```

225 - Lê o cabeçalho de uma gravação em fita cassete.
228 - Lê um byte da fita. O registrador A guardará o byte lido.
231 - Encerra a leitura da fita.
234 - Grava um cabeçalho na fita. Se A=0, o cabeçalho será curto; se A=1, será longo.
237 - Grava um byte na fita. O registrador A deverá conter o byte a ser gravado.
240 - Encerra a gravação.



Muitos endereços e rotinas úteis dos microcomputadores das linhas Apple e TK-2000 foram apresentados no artigo da página 261. Não voltaremos a tratar deles, nem de alguns truques já mencionados, como, por exemplo, o uso de **CALL -151** para ativar o monitor.

O SO do Apple e do TK-2000 divide-se em sistema monitor e rotina de inicialização (*Autostart ROM*). As rotinas citadas aqui são do monitor.

O SO do Apple e do TK-2000 — como o de todos os computadores que utilizam o microprocessador 6502 — tem suas variáveis armazenadas nos primeiros 256 bytes da ROM (página 0). Esses micros não permitem a modificação direta dos registradores do microprocessador 6502 através do BASIC. Assim, as rotinas que requerem parâmetros devem ser usadas com pequenas rotinas em código para leitura e/ou modificação desses registradores.

O comando **CALL** serve para chamar não só essas rotinas mas, também, as que são montadas em código, na RAM. Os endereços podem ser modificados com **POKE** e lidos com **PEEK**.

ROTINAS DO MONITOR

-528 - Imprime na tela o caractere cujo código está no acumulador (registrador A).
-384 - **INVERSE**.
-380 - **NORMAL**.
-198 - Emite um sinal sonoro. No Apple tente também -1059.
-741 - Imprime um cursor piscante e aguarda que se pressione uma tecla, colocando então seu código no acumulador. Também embaralha o gerador de números aleatórios.
-864 - Provoca um atraso de acordo com o valor do acumulador A. No Apple II a duração do atraso é de (26+27*A+5*A*A)/2 microssegundos. No TK-2000 a demora é maior.
-1948 - Determina qual será a cor do gráfico de baixa resolução de acordo com o conteúdo do acumulador.
-1953 - Adiciona o valor 3 ao código da cor atual para gráficos de baixa resolução.
-2048 - Desenha um ponto de baixa resolução. O acumulador define a coordenada vertical, e o registro Y determina a coordenada horizontal.
-2023 - Desenha uma linha horizontal em baixa resolução — coordenada vertical em A, coordenada horizontal inicial em Y e coordenada horizontal final no endereço $2C.
-2008 - Desenha uma linha vertical em baixa resolução — coordenada horizontal em Y, coordenada vertical inicial em A e coordenada vertical final no endereço $2D.
-1998 - Apaga a tela de baixa resolução. No micro Apple, preenche a tela com caracteres @ invertidos, se for chamada no modo texto.

- -1994 - Igual à anterior, só que respeita as quatro linhas de texto no rodapé da tela.
- -1935 - Verifica a cor de uma posição da tela de baixa resolução, usando os mesmos registros que a rotina do endereço -2048. A cor do ponto retornará no acumulador.
- -182 - Grava todos os registros do microprocessador 6502. Usa os endereços $45 a $49.
- -193 - Recupera os registros. Usa os endereços $45 a $49.

TELA DE ALTA RESOLUÇÃO

Os endereços que controlam o tipo de tela no Apple foram muito usados nos programas de *INPUT* e não serão repetidos. Existem, contudo, alguns outros endereços interessantes para gráficos de alta resolução.

- 224/225 - Coordenada X do último ponto plotado.
- 226 - Coordenada Y do último ponto plotado.
- 230 - Indica a página em que o ponto deve ser plotado. O valor 32 corresponde à página 1, e 64, à página 2. Note que o acesso a esse endereço permite que se desenhe na página 2, enquanto se mostra a página 1 e vice-versa.
- 228 - Indica a cor do gráfico. Seus valores estão na tabela apresentada a seguir.

HCOLOR	COR	BYTE 228
0	preto 1	0
1	verde	42
2	violeta	85
3	branco 1	127
4	preto 2	128
5	vermelho	170
6	azul	213
7	branco 2	255

- 62450 - Limpa a tela (alta resolução).
- 62454 - Preenche a tela de alta resolução com o byte 228. Além de permitir a seleção de cores de fundo para seus desenhos, essa rotina produz diversos padrões quando o byte 228 assume valores que não correspondem às cores da tabela anterior.



As rotinas do SO são acessadas pelo comando **EXEC**, que também chama rotinas em código montadas na RAM. Muitas variáveis são obtidas com **PEEK**.

COMUNICAÇÃO COM O CASSETE

Todas as rotinas destinadas ao uso em gravador cassete empregam duas rotinas principais, a saber: **BLKIN** — no endereço 42763 —, que faz a leitura de um bloco de 255 bytes na fita; e **BLKOUT** — endereço 42996 —, que grava um bloco na fita.

Também são de utilidade para o programador BASIC os endereços da página 1, que listamos a seguir:

121 - Fornece o status da porta de entrada e saída do gravador cassete. Pode assumir os valores 0 (porta fechada), 1 (aberta para entrada) e 2 (aberta para saída). Para evitar que erros de saída destruam seu programa, verifique a porta utilizando o comando **PEEK** antes de abrir um arquivo.

144 - Empregado pelo sistema operacional para definir o comprimento do cabeçalho. Se você tem problemas com o volume do gravador, coloque um valor mais alto nesta posição por meio do comando **POKE**.

149/150 - Estas posições contêm o lapso de tempo que se segue a um comando **MOTOR ON**. **POKE 149,0:POKE 150,1** resultará em lapso zero, útil ao se usar **AUDIO ON/OFF** com controle do motor.

PROCESSAMENTO DE TEXTOS

Outra seção do sistema operacional que pode ser de interesse para o programador BASIC é aquela que executa os testes de entrada e saída. Ela inclui entrada do teclado e saída para a tela e a impressora.

Frequentemente, a mensagem "aperte qualquer tecla" surge na tela junto com um laço do tipo:

```
100 IF INKEY$=" " THEN 100
```

Essa linha pode ser substituída por outra bem mais curta: **EXEC 44539**. Se você quiser um cursor piscante use **EXEC 36038**.

Convém anotar ainda os seguintes endereços dessa seção:

135 - Contém o código ASCII da última tecla pressionada.

338 e 345 - Colocando o valor 255 nessas posições, antes do **INKEY$**, obtém-se a auto-repetição das teclas.

282 - É a trava de letras maiúsculas. Você pode forçar as minúsculas colocando um 0 ali. O valor 255 força as maiúsculas e qualquer outro valor desabilita o uso de <**SHIFT**>0.

43304 - Contém a rotina que limpa a tela.

COMUNICAÇÃO COM A IMPRESSORA

Entre as variáveis relacionadas à impressora, são de maior utilidade para o programador aquelas que se encontram nos seguintes endereços:

153 - Determina a distância entre itens separados por vírgulas. O normal é 16.

155 - Para que a função **POS(−2)** tenha um desempenho satisfatório, esse endereço deve conter a largura do papel utilizado (40, 80 ou 132).

330 - Contém o número de caracteres enviados pelo computador para assinalar um final de linha (EOL). Normalmente, apenas um caractere é enviado.

331 a 334 - Contêm os caracteres do **EOL**: CR (13), LF (10), 0, 0. Esses caracteres podem ser alterados para corresponder a um certo tipo de impressora.

328 - Contém o indicador de alimentação automática de linha. Se seu valor for 0, o computador faz o papel avançar automaticamente com o retorno do carro. Qualquer outro valor leva à impressão dos caracteres do **EOL** após a do número de caracteres indicados por 155.

GRÁFICOS

O programador BASIC não tem acesso às rotinas do SO que dizem respeito aos gráficos. Mas existem alguns endereços que, junto com os comandos **PEEK** e **POKE**, podem ser úteis.

182 - **PMODE**: número do modo gráfico.
183/184 - Endereço final da tela.
186/187 - Endereço inicial da tela.
188 - Início da página 1.
200 - Coordenada X do cursor gráfico.
202 - Coordenada Y do cursor gráfico.

OUTRAS ROTINAS INTERESSANTES

EXEC 43486 - Atualiza todos os joysticks; soluciona os problemas que possam ocorrer com o comando **JOYSTK**.

EXEC 40999 - Equivale ao botão **RESET**.

EXEC 41142 - Reinicialização total: equivale a desligar e ligar a máquina novamente.

EXEC 46481 - Executa uma "coleta de lixo" controlada nas variáveis alfanuméricas, evitando as desagradáveis pausas que ocorrem inesperadamente nos programas que as incluem. A quantidade de espaço *string* remanescente pode ser calculada com

```
PEEK(35)*256+PEEK(36)-PEEK(33)*
256-PEEK(34)
```

O valor normal para o espaço string é 200. A não observância desse limite pode causar erros do tipo OS (*out of string*, que quer dizer: fim do espaço para string).

COMO LIDAR COM ARQUIVOS

Abra um canal de comunicação com o drive ou o cassete e mande seus valiosos dados para o arquivo. Saber manipular essa ferramenta é fundamental para certas aplicações.

Qualquer programa que manipule grande quantidade de informação deve armazená-la da maneira apropriada. No artigo publicado à página 128, tratamos do armazenamento de uma pequena lista de telefones em linhas **DATA**. O programa ali apresentado permite ao usuário procurar e imprimir qualquer item da lista. Porém, para qualquer modificação na lista, é necessário interromper a execução do programa e editar as linhas **DATA**. Além disso, os dados só podem ser gravados junto com o programa que os manipula.

Essa dependência dos dados ao programa muitas vezes é indesejável. Para evitá-la, convém guardar os dados em fita cassete ou disco flexível. A recuperação das informações é tão rápida quanto no caso das linhas **DATA**, mas a alteração e a edição de dados tornam-se extremamente mais fáceis. Para efetuá-las, o usuário não precisa saber nada a respeito do programa.

Como em muitos programas aplicativos listados em *INPUT*, os dados são armazenados em arquivos, na fita ou disquete. O banco de dados, o orçamento doméstico, a agenda eletrônica e a planilha são exemplos de programas que envolvem grande quantidade de informações. Se observarmos suas listagens com atenção, poderemos identificar as linhas que tratam da gravação e leitura dos dados. Elas ocupam uma porção mínima do programa, se comparadas às demais seções, encarregadas, entre outras coisas, de reservar espaço na memória, estabelecer a comunicação com o usuário e manipular os dados.

Quando armazenamos os dados à parte, as possibilidades de utilização do programa aumentam bastante, estendendo-se à criação e alteração de vários arquivos diferentes, com quantidades diversas de registros e campos.

Empregamos em nosso programa o método de armazenagem seqüencial, cujo princípio é muito simples. Como você verá, ele permite que os dados sejam convertidos a um formato que se adapta a diversos programas, e até mesmo a computadores diferentes.

O arquivo seqüencial caracteriza-se pela armazenagem dos dados em série, dispostos um após o outro e separados apenas por um byte. Toda a informação deve estar na memória antes de ser transferida ao arquivo.

O primeiro passo para a criação de um arquivo seqüencial consiste na abertura de um canal de comunicação com o dispositivo que vai armazenar os dados — cassete ou drive. Isso é feito por intermédio do comando **OPEN**, numa sintaxe que varia conforme o tipo do microcomputador. Para conhecer os comandos mais utilizados, volte ao artigo da página 688.

Aqui, explicamos em detalhe as técnicas envolvidas na criação de arquivos, de modo que você poderá incorporá-las facilmente a seus programas.

CRIANDO UM ARQUIVO

O programa a seguir mostra como criar uma versão mais sofisticada da lista de telefones mencionada anteriormente. Ao utilizá-la, você poderá modificar, sem maiores dificuldades, as mensagens dos comandos **INPUT** e o formato das matrizes, para que aceitem outro tipo de informação.

Os dados são fornecidos nesta ordem: nome, sobrenome e número do telefone. Como números de telefone podem conter espaços, hífens e parênteses, convém dar-lhes a forma de cadeia de caracteres. Se você quiser alterar o programa para lidar com outros tipos de informação, poderá usar também matrizes numéricas e alfanuméricas. Um arquivo recebe dados em qualquer formato e em qualquer ordem. O importante é que a leitura se faça nessa mesma ordem e que os dados sejam colocados no tipo adequado de variável.

Embora seja possível armazenar informações diretamente no arquivo, é mais conveniente que coloquemos os dados em uma matriz e depois guardemos toda a matriz no arquivo.

A primeira parte do programa tem um pequeno laço que possibilita a entrada dos dados. Digite quantos nomes e números quiser — o limite é a dimensão da matriz, especificada na linha 10. Modifique a dimensão, caso precise de um número de itens superior a cinqüenta. Quando tiver entrado todos os dados, digite **ENTER** ou **RETURN** e o laço de entrada será interrompido.

A segunda parte do programa — da linha 100 em diante — grava os dados no arquivo. Daremos explicações referentes a seu funcionamento nas várias versões do programa.

S

```
10 DIM A$(50,15): DIM B$(50,
15): DIM T$(50,12): DIM N(1)
20 LET N=0
30 LET N=N+1
40 INPUT "PRIMEIRO NOME ";A$(
N)
50 INPUT "SEGUNDO NOME ";B$(N
)
60 INPUT "NUMERO DO TELEFONE
";T$(N)
70 IF A$(N)<>"
" AND N<50 THEN  GOTO 30
80 CLS : PRINT "SALVANDO DADO
S AGORA"
100 SAVE "CONT" DATA N()
110 SAVE "P.NOMES" DATA A$()
120 SAVE "S.NOMES" DATA B$()
130 SAVE "N.TELEF." DATA T$()
140 PRINT "DADOS GRAVADOS"
150 STOP
```

Para a gravação, o Spectrum utiliza o comando **SAVE** seguido do nome do arquivo. Note que cada matriz é gravada como um arquivo distinto, recebendo um nome especial. A instrução **DATA** seguida do nome da matriz deve vir depois do nome do arquivo.

MSX

```
10 DIM A$(50),B$(50),T$(50)
30 N=N+1
40 INPUT "PRIMEIRO NOME";A$(N)
50 INPUT "SEGUNDO NOME";B$(N)
60 INPUT "TELEFONE";T$(N)
70 IF A$(N)<>"" AND N<50 THEN 3
0
80 CLS:PRINT "GRAVANDO OS DADOS
"
100 OPEN "CAS:ARQ" FOR OUTPUT A
S #1
110 PRINT #1,N
120 FOR L=1 TO N
130 PRINT #1,A$(L),B$(L),T$(L)
140 NEXT
150 CLOSE #1
160 PRINT "DADOS GRAVADOS"
170 END
```

Os dados entrados na primeira parte do programa são colocados em três va-





Como gravar dados em fita cassete no Apple e no TK-2000?

Para contornar os problemas de arquivamento no Apple e no TK-2000, a melhor solução é comprar um drive de discos flexíveis. Mas existem algumas alternativas — nenhuma milagrosa, podemos adiantar.

Esses micros só permitem a gravação de uma variável indexada unidimensional e não montam na fita um arquivo propriamente dito. O leitor pode achar que isso já é suficiente para algumas aplicações. Porém, a lentidão e o grande espaço que uma pequena variável ocupa na fita certamente logo vão desencorajar os mais afoitos.

Em nossa opinião, a saída mais conveniente para quem não pode adquirir um drive é colocar os dados em um *buffer* no alto da memória, usando o comando **POKE**. Em seguida, basta entrar no monitor de linguagem de máquina — com **CALL -151** — e gravar o buffer por intermédio do comando **W** do monitor.

riáveis indexadas: **A$( )**, **B$( )** e **T$( )** Na linha 100, abrimos um arquivo para a gravação de dados em fita cassete por intermédio da instrução **OPEN "CAS: nome do arquivo" FOR OUTPUT AS # número do arquivo**.

A instrução **PRINT #1** significa "imprima o próximo item no arquivo". Assim, a linha 110 grava o valor de N — total de números e nomes da lista —, para que seja usado posteriormente na leitura do arquivo.

O laço entre as linhas 120 e 140 utiliza a mesma instrução **PRINT #1** para guardar o conteúdo das três variáveis indexadas. A instrução **CLOSE #1**, por sua vez, fecha o arquivo.



```
10  DIM A$(50),B$(50),T$(50)
20 D$ =  CHR$ (4)
30 N = N + 1
40  INPUT "PRIMEIRO NOME ";A$(N
)
50  INPUT "SEGUNDO NOME ";B$(N)
60  INPUT "TELEFONE ";T$(N)
70  IF A$(N) <  > "" AND N < 50
 THEN 30
80  HOME : PRINT "GRAVANDO OS D
ADOS"
100  PRINT D$;"OPEN ARQUIVO"
110  PRINT D$;"WRITE ARQUIVO"
120  PRINT N
130  FOR L = 1 TO N
140  PRINT A$(L): PRINT B$(L):
PRINT T$(L)
150  NEXT L
160  PRINT D$;"CLOSE ARQUIVO"
170  PRINT "DADOS GRAVADOS"
180  END
```

Trataremos apenas da gravação em disquete, pois a armazenagem de arquivos em fita no Apple e no TK-2000 é muito precária. Como você deve se lembrar, para controlar o drive sem sair do BASIC Applesoft utilizamos o caractere de controle **CHR$(4)**, que foi colocado na variável **D$** (linha 5).

A primeira parte do programa, que cuida da entrada dos dados, coloca-os em três variáveis indexadas: **A$( )**, **B$( )** e **T$( )**. Um arquivo é aberto na linha 100 através do caractere de controle, seguido do comando **OPEN** e do nome do arquivo (entre aspas).

O comando **WRITE**, na linha 110, permite a gravação de dados no arquivo que foi aberto. Todos os comandos **PRINT** que aparecem em seguida escrevem no arquivo e não na tela. O comando **CLOSE** fecha o arquivo e devolve a função original ao **PRINT**.



```
10 DIM A$(50),B$(50),T$(50)
30 N=N+1
40 INPUT"PRIMEIRO NOME ";A$(N)
50 INPUT"SEGUNDO NOME ";B$(N)
60 INPUT"NUMERO DO TELEFONE ";T
$(N)
70 IF A$(N)<>"" AND N<50 THEN 3
0
80 CLS:PRINT"SALVANDO DADOS AGO
RA"
```

Use esta seção para arquivos em fita cassete:

```
100 OPEN "O",#-1,"ARQUIVO"
110 PRINT #-1,N
120 FOR L=1 TO N
130 PRINT #-1,A$(L),B$(L),T$(L)
140 NEXT
150 CLOSE #-1
160 PRINT"DADOS GRAVADOS"
170 END
```

Utilize esta seção para arquivos em disquete:

```
100 OPEN "O",#1,"ARQUIVO"
110 PRINT #1,N
120 FOR L=1 TO N
130 PRINT #1,A$(L),B$(L),T$(L)
140 NEXT
150 CLOSE #1
160 PRINT"DADOS GRAVADOS"
170 END
```

A primeira seção do programa, que trata da entrada dos dados, é comum aos dois tipos de dispositivo, ao contrário da segunda seção, incumbida de gravar os dados.

Na versão para fita cassete, um arquivo é aberto pela instrução **OPEN "O"** que estabelece uma linha de comunicação com o dispositivo de armazenamento para a saída de dados. Ela é seguida por um #−1, que diz ao computador que se trata de gravador cassete, e pelo nome do arquivo entre aspas.

O comando **PRINT # −1** significa "imprima o próximo valor no arquivo". Assim, a linha 110 grava o valor de N — o número de telefones da lista. Para o processo de leitura, que descrevemos a seguir, é importante que esse valor encabece o arquivo.

A linha 130 escreve cada um dos itens no arquivo — observe que eles são separados por meio de vírgulas. Finalmente, o comando **CLOSE # −1** fecha o arquivo (linha 150).

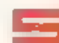
A criação de arquivos em discos flexíveis é muito semelhante, empregando os mesmos comandos. A única diferença é o número utilizado após o sinal #. **OPEN "O", #1**, seguida do nome do arquivo entre aspas, abre um arquivo; **PRINT #1** guarda um item nele e **CLOSE #1** trata de fechá-lo.

Como você pode notar, na linha 130 os itens estão separados por ponto e vírgula. Embora se permita o uso de vírgulas, aquele é o modo mais econômico de armazenar dados no disco.

Os disquetes admitem a criação de mais de um arquivo ao mesmo tempo, pois, ao contrário do gravador cassete, o drive é capaz de procurar a porção adequada para ali gravar os dados. Assim, podemos numerar os arquivos com #1, #2, #3 e assim por diante, enquanto o cassete é sempre numerado com # −1 — um **PRINT # −2** mandará o dado para a impressora.

A LEITURA DO ARQUIVO

O programa que recupera as informações gravadas no arquivo é exatamente o inverso do programa anterior. Ele imprime os dados lidos na tela para que possamos conferi-los.



```
200 CLEAR
210 DIM N(1): LOAD "CONT" DATA
 N()
```

```
215 DIM A$(N(1),15): DIM B$(N(
1),15): DIM T$(N(1),12)
220 LOAD "P.NOMES" DATA A$()
230 LOAD "S.NOMES" DATA B$()
240 LOAD "N.TELEF." DATA T$()
250 FOR L=1 TO N(1)
260 PRINT A$(L),B$(L),T$(L)
270 NEXT L
```

Para utilizar esse programa, rebobine a fita até o começo do arquivo gravado pelo primeiro programa. Digite **RUN 200** para recuperar os dados; não rode o programa inteiro novamente.

A linha 200 apaga os antigos valores das variáveis, para que só os obtidos no arquivo sejam impressos. A linha 210 dimensiona uma nova variável **N( )** com um elemento. Ela será usada para recuperar o valor de N gravado anteriormente. As outras variáveis são dimensionadas de acordo com o valor de **N( )**. Conhecendo o número exato de elementos do arquivo, não tentaremos ler além do seu final.

O resto do programa é fácil de entender: simplesmente substituímos os **SAVE** do primeiro programa por **LOAD**. É preciso fechar um arquivo aberto para gravação, mas não para leitura.



```
200 CLEAR 2000:OPEN"CAS:ARQUIVO
"FOR INPUT AS #1
210 INPUT#1,N:DIM A$(N),B$(N),T
$(N)
220 FOR L=1 TO N
230 INPUT #1,A$(L),B$(L),T$(L)
235 PRINT A$(L);B$(L);T$(L)
240 NEXT L
250 CLOSE #1
```

Para usar esse programa, rebobine a fita até o começo do arquivo gravado pelo primeiro programa. Digite **RUN 200** para ler o arquivo; não rode o programa inteiro novamente.

A linha 200 apaga as variáveis originais; dessa maneira, o programa imprime só aquilo que obtiver no arquivo. Este é aberto para leitura pela instrução **OPEN "CAS:" FOR OUTPUT AS**, seguida do valor adequado.

A linha 210 lê a quantidade de nomes armazenada no arquivo — **N** — e utiliza esse valor para dimensionar as variáveis. **N** também controla o laço de leitura do arquivo, para que ele seja recuperado até o final. Note que os valores lidos são colocados nas variáveis na ordem em que foram gravados, separados pelo mesmo tipo de sinal — no caso da fita, a vírgula. O arquivo deve ser fechado após ser lido.



```
200 D$ =  CHR$ (4): PRINT D$;"O
PEN ARQUIVO"
210  PRINT D$;"READ ARQUIVO"
220  INPUT N: DIM A$(N),B$(N),T
$(N)
230  FOR L = 1 TO N
240  INPUT A$(L): PRINT A$(L)
250  INPUT B$(L): PRINT B$(L)
260  INPUT T$(L): PRINT T$(L)
270  NEXT L
280  PRINT D$;"CLOSE ARQUIVO"
```

A linha 200 é encarregada de apagar os antigos valores das variáveis; dessa maneira, o programa imprime somente os valores que obtiver no arquivo. Este é aberto para leitura pela instrução **OPEN** seguida de **READ**.

A linha 210 lê a quantidade de nomes armazenada no arquivo — **N** — e utiliza esse valor para dimensionar as variáveis. **N** também tem como função controlar o laço de leitura do arquivo, para que ele seja recuperado até o final. Note que os valores lidos são colocados nas variáveis na mesma ordem em que foram gravados, separados pelo mesmo tipo de sinal — no caso da fita, a vírgula. O arquivo deve ser fechado após ser lido.



Para fita cassete, use esta seção:

```
200 CLEAR 2000:OPEN"I",#-1,"ARQ
UIVO"
210 INPUT#-1,N:DIM A$(N),B$(N),
T$(N)
220 FOR L=1 TO N
230 INPUT # 1,A$(L),B$(L),T$(L)
235 PRINT A$(L);B$(L);T$(L)
240 NEXT L
250 CLOSE #-1
```

No caso de discos flexíveis, recorra a esta outra:

```
200 CLEAR 2000:OPEN "O",#1,"ARQ
UIVO"
210 INPUT#1,N:DIM A$(N),B$(N),T
$(N)
220 FOR L=1 TO N
230 INPUT #1,A$(L),B$(L),T$(L)
235 PRINT A$(L);B$(L);T$(L)
240 NEXT L
250 CLOSE #1
```

Para executar essa parte do programa, rebobine a fita até o início do arquivo. Digite o comando **RUN 200** para recuperar os dados; não rode o programa inteiro novamente.

A linha 200 apaga os antigos valores das variáveis; dessa maneira, o programa imprime apenas os valores que obtiver no arquivo. Este é aberto para leitura pela instrução **OPEN "I"** seguida do valor adequado — **# −1** para fita e **#1** para disquete.

A linha 210 lê a quantidade de nomes armazenada no arquivo — **N** — e utiliza esse valor para dimensionar as variáveis. **N** também controla o laço de leitura do arquivo, para que ele seja recuperado até o final. Note que os valores lidos são colocados nas variáveis na ordem em que foram gravados, separados pelo mesmo tipo de sinal — no caso da fita, a vírgula; no do disco, ponto e vírgula. O arquivo deve ser fechado após ser lido.



FIM DE ARQUIVO

O último programa usou a variável N para controlar a leitura do arquivo. Contudo, nem sempre é possível saber de antemão quantos itens devem ser lidos. Um modo de contornar o problema é usar um marcador para sinalizar o fim do arquivo. O TRS-Color e o MSX colocam automaticamente um sinal desse tipo — **EOF** — no final, sempre que uma instrução **CLOSE** é encontrada.

Assim os usuários dos microcomputadores TRS-Color e MSX podem modificar o programa anterior para que ele leia o arquivo desde o início, parando sempre que encontrar o sinal que indica fim de arquivo — **EOF**.

Listamos o programa a partir da linha 100 para que você possa compará-lo à versão já dada e escolher o método que considerar melhor.

Modifique as linhas conforme a listagem abaixo e, depois, tente gravar e ler o arquivo novamente.



```
100 OPEN "CAS:ARQUIVO"FOR OUTPU
T AS #1
110 N=0
120 N=N+1
130 PRINT #1,A$(N),B$(N),T$(N)
140 IF A$(N)<>"" AND N<50 THEN
120
150 CLOSE #1
160 PRINT "DADOS GRAVADOS"
170 END
200 CLEAR 2000;DIM A$(50),B$(50
),T$(50):OPEN "CAS:ARQUIVO" FOR
 INPUT AS #1
230 N=N+1:INPUT #1,A$(N),B$(N),
T$(N)
235 PRINT A$(N);B$(N);T$(N)
240 IF EOF(1)=0 THEN 230
250 CLOSE #1
```



Apague as linhas 210 e 220 e modifique as demais. Para a fita:

```
100 OPEN "O",#-1,"ARQUIVO"
110 N=0
120 N=N+1
130 PRINT #-1,A$(N),B$(N),T$(N)
140 IF A$(N)<>"" AND N<50 THEN
120
150 CLOSE #-1
160 PRINT "DADOS GRAVADOS"
170 END
200 CLEAR 2000:DIM A$(50),B$(50
),T$(50):OPEN "I",#-1,"ARQUIVO"
230 N=N+1:INPUT #-1,A$(N),B$(N)
,T$(N)
235 PRINT A$(N):B$(N):T$(N)
240 IF EOF(-1)=0 THEN 230
250 CLOSE #-1
```

Para o disquete:

```
100 OPEN "O",#1,"ARQUIVO"
110 N=0
120 N=N+1
130 PRINT #1,A$(N),B$(N),T$(N)
140 IF A$(N)<>"" AND N<50 THEN
120
150 CLOSE #1
160 PRINT "DADOS GRAVADOS"
170 END
200 CLEAR 2000:DIM A$(50),B$(50
),T$(50):OPEN "O",#1,"ARQUIVO"
230 N=N+1:INPUT #1,A$(N),B$(N),
T$(N)
235 PRINT A$(N);B$(N);T$(N)
240 IF EOF(1)=0 THEN 230
250 CLOSE #1
```

A linha 240 de ambos os programas verifica a ocorrência de um sinal de fim de arquivo — **EOF**.



OUTROS SINALIZADORES

Para indicar o final do arquivo, os usuários dos microcomputadores da linha Apple podem colocar ali um sinalizador arbitrário. Já no caso do Spectrum, detectamos o fim do arquivo de outra maneira: após entrar todos os dados no laço inicial, digitamos uma senha arbitrária. Esta pode ser **ZZZ**, **−999**, a palavra **FIM** ou qualquer outra coisa que normalmente não se inclui entre os dados de um arquivo.

Altere as próximas linhas e veja como utilizar um sinalizador desse tipo. Ao digitar os dados do arquivo, entre a palavra **FIM** após o último número de telefone e **ENTER** ou **RETURN** após as outras duas solicitações.



```
70 IF A$(N)<>"FIM" AND N<50 THE
N 30
140 IF A$(N)<>"FIM" AND N<50 TH
EN 120
240 IF A$(N)<>"FIM" AND N<50 TH
EN 230
```



```
10  DIM A$(50),B$(50),T$(50)
20 D$ =  CHR$ (4)
30 N = N + 1
40  INPUT "PRIMEIRO NOME ";A$(N
)
50  INPUT "SEGUNDO NOME ";B$(N)
60  INPUT "TELEFONE ";T$(N)
70  IF A$(N) <  > "FIM" AND N <
 50 THEN 30
80  HOME : PRINT "GRAVANDO OS D
ADOS"
100  PRINT D$;"OPEN ARQUIVO"
110 N = 0
120 N = N + 1
130  PRINT D$;"WRITE ARQUIVO"
140  PRINT A$(N): PRINT B$(N):
PRINT T$(N)
150  IF A$(N) <  > "FIM" AND N
< 50 THEN 120
160  PRINT D$;"CLOSE ARQUIVO"
170  PRINT "DADOS GRAVADOS"
180  END
200 D$ =  CHR$ (4): PRINT D$;"O
PEN ARQUIVO"
210  PRINT D$;"READ ARQUIVO"
220  DIM A$(50),B$(50),T$(50)
225 L = 0
230 L = L + 1
240  INPUT A$(L): PRINT A$(L)
250  INPUT B$(L): PRINT B$(L)
260  INPUT T$(L): PRINT T$(L)
270  IF A$(L) <  > "FIM" AND L
< 50 THEN  GOTO 230
280  PRINT D$;"CLOSE ARQUIVO"
```



```
70 IF A$(N)<>"FIM" AND N<50 THE
N 30
140 IF A$(N)<>"FIM" AND N<50 TH
EN 120
240 IF A$(N)<>"FIM" AND N<50 TH
EN 230
```

O USO DOS DADOS

Armazenados em um arquivo, ou seja, fora de um programa BASIC, os dados podem ser utilizados por vários programas diferentes. Qualquer uma das rotinas de ordenação apresentadas nos artigos das páginas 468 e 738, por exemplo, servem para ordenar alfabeticamente nossa lista telefônica. Uma vez feito isso, podemos localizar rapidamente determinado item da lista, por meio de uma busca binária, como mostra o artigo da página 930.

PROGRAMA PARA TESTE DO VÍDEO

- A IMAGEM ESTÁ CENTRADA?
- DISTORÇÕES JUNTO ÀS BORDAS
- DEFINIÇÃO DA IMAGEM
- RAZÃO DE ASPECTO
- DEFINIÇÃO DAS CORES

Se você sentir a vista cansada após fixar por algum tempo o vídeo do micro, talvez seja a hora de providenciar um outro monitor. Use este programa para testar a qualidade da imagem.

Até pouco tempo atrás, o preço de um monitor de vídeo colocava esse periférico totalmente fora do alcance da maioria dos proprietários de micros, obrigando-os a utilizar aparelhos comuns de TV como saída de vídeo.

Os televisores apresentam vários inconvenientes: a imagem pisca e oscila continuamente de modo imperceptível, o que provoca dores de cabeça e cansaço visual após um certo tempo. Outro problema é causado pela baixa resolução da tela, que resulta em perda de detalhes e definição da imagem em textos e gráficos.

Entretanto, o custo relativo dos monitores profissionais de vídeo tem caído bastante nos últimos anos, tornando esse periférico cada vez mais acessível. E, se você ainda não tem um, vale a pena ir pensando em adquiri-lo, pois a qualidade da imagem é bem superior à de um televisor.

No artigo da página 851, descrevemos as características técnicas que diferenciam um televisor de um monitor e falamos dos aspectos que devem ser considerados na escolha de um bom monitor. Não há dúvida, porém, de que o melhor critério para essa seleção ainda é o "Teste de São Tomé" — ou seja, ver como fica a imagem no vídeo, quando o conectamos ao nosso computador. Como há problemas de compatibilidade entre monitores e micros, é fundamental que você use o seu modelo na hora desse teste.

O programa listado neste artigo foi projetado para testar as características do vídeo em diferentes linhas de computadores. A versão para o Spectrum mostra um único padrão visual na tela; já as versões para o MSX, Apple, TK-2000 e TRS-Color mostram duas ou três telas sucessivas, cada qual projetada para testar um aspecto do funcionamento do vídeo.

Os padrões visuais variam de computador para computador, mas contêm basicamente os elementos utilizados em testes como os que são transmitidos em horas mortas pelas emissoras de TV: círculos e linhas espaçadas a curta distância, para testar o poder de definição de tela; padrões colocados nos cantos da tela, para testar distorções marginais da imagem, e barras ou faixas de todas as cores disponíveis.

Ao rodar o programa de teste, observe primeiro se a imagem está bem centrada. Em alguns computadores, o gerador de vídeo não deixa margem na tela, e, se ela estiver mal ajustada, haverá uma perda de gráficos e textos junto à borda. Muitos monitores e televisores dispõem de botões giratórios de ajuste de centralização e expansão de imagem, localizados atrás do aparelho. Você poderá usá-los para conseguir um resultado melhor.

Verifique em seguida se as linhas dos retângulos junto às bordas da imagem não sofrem uma grande distorção (as linhas retas se transformam em arcos convexos: quanto menor o raio destes, maior a probabilidade de que o monitor seja de baixa qualidade).

Todas as linhas aparecem nitidamente separadas umas das outras? Se isso não ocorrer, há problemas de definição na imagem, o que poderá prejudicar a legibilidade de textos. Em muitos monitores e televisores, a resolução é melhor no sentido horizontal do que no vertical, ou vice-versa, devido à maneira como os pontos coloridos de fósforo são dispostos na tela. Monitores de boa qualidade têm resolução igual em todas as direções.

Os círculos que o programa desenha não são perfeitos, tendendo a assumir uma forma ovalada? Este é um bom teste para uma das características dos vídeos de boa qualidade, a chamada *razão de aspecto*: para obtê-la, dividimos o diâmetro transversal do círculo pelo diâmetro vertical. Idealmente, é claro, o resultado deverá ser 1.

Verifique também se as linhas aparecem traçadas com nitidez ou se as bordas e cantos parecem borrados. Um problema desse tipo pode prejudicar a legibilidade de um texto na tela.

Se o monitor ou TV for colorido, examine as cores das barras (ou das letras, no programa para o TRS-Color) na segunda parte do teste. Os cantos e bordas das barras coloridas devem ser bem definidos, e a cor de uma barra não deve extravasar para outra. Como a densidade, brilho e a saturação das cores podem variar de vídeo para vídeo, experimente mudar os controles do aparelho, até obter a melhor combinação. Lembre-se de que diferentes ajustes de cor costumam ser necessários, conforme a aplicação: texto puro, por exemplo, requer pouca cor e brilho médio; um jogo exige muita cor e brilho.

Os programas para o MSX e o TRS-Color permitem variar a cor de fundo na tela pressionando seguidamente a tecla **<ENTER>**, na última parte do programa. Para interrompê-lo, pressione **<BREAK>** ou **<CTRL><STOP>**.

No Apple e no TK-2000, os caracteres de texto não podem ser coloridos. Assim, para testar a qualidade do vídeo quanto a este aspecto, interrompa o programa e liste-o na tela. Observe se as letras aparecem em branco puro e se não há nenhum efeito de "arco-íris" (letras multicoloridas e borradas).

Se você conhece um pouco de programação gráfica em BASIC, não terá nenhuma dificuldade em modificar os programas aqui apresentados para torná-los mais completos ou atraentes. Procure alterar a combinação de cores de frente e de fundo, para verificar o efeito, gere mais uma tela de textos e caracteres gráficos etc.



```
10 PMODE 4,1:PCLS:SCREEN 1,1
20 FOR K=0 TO 1:FOR J=0 TO 15 S
TEP K+2
30 LINE (K*16+J,K*16+J)-(255-K*
16-J,191-K*16-J),PSET,B
40 NEXT J,K
50 FOR K=100 TO 154 STEP 2:LINE
(K,85)-(K,105),PSET:LINE(113,K-
32)-(141,K-32),PSET·NEXT
60 FOR K=1 TO 100 STEP 10:CIRCL
E(127,95),K,5:NEXT
70 IF INKEY$="" THEN 70
80 PMODE 1,1:PCLS:SCREEN 1,0
90 FOR J=0 TO 191 STEP 48:COLOR
 J/48+1
```

Padrões de teste de vídeo utilizados em diferentes computadores.

```
100 LINE(0,J)-(255,J+47),PSET,B
F:NEXT
110 FOR J=0 TO 255 STEP 63:FOR
K=0 TO 63 STEP 4:COLOR J/63+1
120 LINE(K+J,0)-(K+J,191),PSET
130 NEXT K,J
140 IF INKEY$="" THEN 140
150 S=1-S:SCREEN 1,S:GOTO 140
```



```
 5 FOR p=0 TO 7: FOR i=0 TO 7
 : IF p=i THEN GOTO 200
 6 PAPER p: INK i: BORDER 0:
 CLS
10 FOR n=0 TO 12 STEP 2: PLOT
n,n: DRAW 0,175-2*n: DRAW 255
-2*n,0: DRAW 0,-(175-2*n):
DRAW -(255-2*n),0: NEXT n
20 FOR n=10 TO 20 STEP 2:
CIRCLE INK i;128,85,n: NEXT
n
30 FOR n=5 TO 7: PRINT INK i
;AT n,8;" ";AT n+10,8;" ":
NEXT n
40 PRINT AT 4,8;: FOR m=0 TO
1: FOR n=0 TO 7: PRINT PAPER
n;" ";: NEXT n: NEXT m
50 PRINT AT 18,8;: FOR m=0 TO
1: FOR n=0 TO 7: PRINT PAPER
n;" ";: NEXT n: NEXT m
55 PRINT AT 2,4;: FOR n=0 TO
2: FOR m=0 TO 7: PRINT INK m
; PAPER p;CHR$ (64+n*7+m);:
NEXT m: NEXT n
57 PRINT AT 19,4;: FOR n=0 TO
2: FOR m=0 TO 7: PRINT
BRIGHT 1; INK m; PAPER p;CHR$
(64+n*7+m);: NEXT m: NEXT n
60 FOR n=0 TO 7: BORDER n:
PAUSE 50: NEXT n: PAUSE 50
200 NEXT i: NEXT p
```



```
10 S=S+1:COLOR 15,S,S:SCREEN 2
15 CL=15:IF S=15 THEN CL=1
20 FOR K=0 TO 1:FOR J=0 TO 15 S
TEP K+2
30 LINE(K*16+J,K*16+J)-(255-K*1
6-J,191-K*16-J),CL,B
40 NEXT J,K
50 FOR K=100 TO 154 STEP3:LINE(
K,85)-(K,105),CL:NEXT
55 FOR K=65 TO 135 STEP3:LINE(1
40,K)-(114,K),CL:NEXT
60 FOR K=1 TO 100 STEP 10:CIRCL
E(127,95),K,CL:NEXT
70 IF INKEY$="" THEN 70
80 COLOR 15,S,S:SCREEN 2
90 FOR J=0 TO 191 STEP 13
100 LINE(0,J)-(255,J+11),J/13+1
,BF:NEXT
110 IF INKEY$="" THEN 110
130 GOTO 10
```



```
10  HGR2 :LX = 279:LY = 180
20  FOR X0 = 0 TO 20 STEP 4
25  LET Y0 = X0:LX = LX - 10:LY
 = LY - 10
30  HPLOT X0,Y0 TO X0 + LX,Y0 T
O X0 + LX,Y0 + LY TO X0,Y0 + LY
 TO X0,Y0
40  NEXT X0
50  FOR Y = 60 TO 100 STEP 2
55  HPLOT 110,Y TO 150,Y
57  HPLOT 50,Y TO 70,Y: HPLOT 1
90,Y TO 210,Y
60  NEXT Y
65  FOR X = 95 TO 165 STEP 3
70  HPLOT X,68 TO X,93
75  NEXT X
85  FOR R = 10 TO 60 STEP 10
86  HPLOT 130 + R,80
87  FOR A = 0 TO 6.4 STEP .1
90  HPLOT  TO 130 + R *  COS (A
),80 + R *  SIN (A)
95  NEXT A: NEXT R
100  GET A$
105  TEXT : HOME
110  GR :C = 0
120  FOR I = 0 TO 36 STEP 3
130  LET C = C + 1: COLOR= C: V
LIN 1,15 AT I
135  VLIN 1,15 AT I + 1: VLIN 1
,15 AT I + 2
140  NEXT I:C = 0
145  FOR J = 18 TO 36 STEP 2
150  LET C = C + 1: COLOR= C
155  HLIN 0,38 AT J: HLIN 0,38
AT J + 1
160  NEXT J
190  GET A$
200  TEXT : HOME : END
```

CONTROLE A ENTRADA DE DADOS

"Envenene" seus programas de entrada de dados com uma rotina de grande versatilidade. Ela substitui com vantagem o comando INPUT, superando várias de suas limitações.

O comando **INPUT** satisfaz a maioria das necessidades de programação de entrada de dados. Em certas situações, porém, a utilização desse comando apresenta alguns sérios inconvenientes. Entre eles, podemos citar:

- toda vez que o programa executa um comando **INPUT**, o cursor muda de linha na tela quando o usuário pressiona a tecla **<ENTER>**;
- um ponto de interrogação é sempre colocado na tela (nos micros pertencentes às linhas TRS-80, TRS-Color e MSX), ou força a entrada na última linha (caso dos micros da linha Sinclair: ZX-81 e Spectrum);
- todos os caracteres digitados aparecem na tela: é impossível ocultá-los (como se faz, por exemplo, na entrada de uma senha secreta);
- é necessário pressionar **<ENTER>** para terminar a entrada;
- não existe nenhum controle sobre os caracteres que estão sendo digitados. Na linha TRS, por exemplo, se pressionarmos a tecla de alimentação de linha (representada pela flecha para baixo), prejudicaremos a formatação de uma tela de entrada.

É possível contornar todos os problemas mencionados recorrendo a uma rotina especial de entrada, que substitui o comando **INPUT**.

Neste artigo, ensinaremos como desenvolver essa rotina usando a função **INKEY$**, disponível no BASIC dos micros pertencentes às linhas Sinclair, TRS e MSX, ou o comando **GET**, nos micros das linhas Apple e TK-2000.

UMA ROTINA SIMPLES

Inicialmente, vamos programar a rotina de entrada de dados da forma mais simples possível, para que você possa entendê-la sem dificuldade:



```
1000 LET S$=""
1010 LET C$=INKEY$
1020 IF C$="" THEN GOTO 1010
1022 LET C=ASC(C$)
1025 IF C=13 THEN RETURN
1030 LET S$=S$+C$
1040 PRINT C$;
1060 GOTO 1010
```



Modifique as seguintes linhas, para o micro ZX-81:

```
1022 LET C=CODE C$
1025 IF C=118 THEN RETURN
```



Suprima a linha 1020 e substitua a linha 1010, para o programa funcionar no Apple e no TK-2000:

```
1010 GET C$
```

Teste a rotina, acrescentando a este pequeno programa:



```
10 PRINT
90 PRINT "ENTRE: ";
100 GOSUB 1000
110 PRINT
120 PRINT S$
130 GOTO 10
```



Para evitar problemas de "estouro" da área alfanumérica, acrescente esta linha para o TRS-80 e TRS-Color:

```
5 CLEAR 1000
```

FUNCIONAMENTO DA ROTINA

A linha 1000 inicializa a cadeia de saída, **S$**, que conterá tudo o que for digitado. As linhas 1010 e 1020 verificam se alguma tecla foi pressionada. Se a resposta for negativa, o programa continua esperando. Caso contrário, o programa prossegue para a linha 1030, que testa se a tecla pressionada foi **<ENTER>** ou **<RETURN>**, que assinalam o fim da entrada de dados. Se foi outra tecla qualquer, ela é concatenada à cadeia de saída, na linha 1030, e impressa na linha 1040. Finalmente, o programa retorna à linha 1010, para aguardar a próxima tecla.

LIMITE PARA A ENTRADA

Ao retornar da rotina, o cursor continua na mesma linha de entrada. A linha 100 força sua mudança para a linha seguinte, mas pode ser retirada, se necessário — por exemplo, se você estiver escrevendo um programa de formatação de tela de entrada, com vários campos da mesma linha.

O programa principal pode ser modificado de modo a colocar o cursor num ponto determinado da tela, antes de chamar a sub-rotina da linha 1000. Acrescente as seguintes linhas para verificar o efeito obtido:



APLICAÇÕES PARA A ROTINA

A rotina apresentada neste artigo tem variadas aplicações. Usando-a criativamente, você poderá trabalhar com diversas funções impossíveis de se obter com o BASIC. Essa rotina equivale, de fato, a um poderoso comando **LINE INPUT** que alguns interpretadores possuem.

Em programas educativos e de jogos, você pode incluir uma função de atraso máximo de tempo dentro da rotina de entrada — ou seja, a cada repetição do laço de varredura (apenas para as versões com o comando **INKEY$**), uma variável **T** de tempo pode ser incrementada. Ao sair da rotina, o programa tem acesso ao **T** total, informando ao usuário o tempo que levou para entrar os dados.

Outra aplicação interessante consiste em sair automaticamente da rotina quando um tempo máximo estipulado para a resposta foi ultrapassado — o que é últil em jogos de rapidez de raciocínio, por exemplo.

```
80 PRINT AT 10,10;
```

```
80 PRINT @ 128,"";
```

```
80 LOCATE 10,10
```

```
80 HTAB 10:VTAB 10
```

APERFEIÇOAMENTOS

Como está, a rotina oferece poucos recursos e não apresenta vantagens de ordem significativa em relação ao comando **INPUT**. Mas existem várias maneiras de aperfeiçoá-la. Podemos, por exemplo, estabelecer um limite máximo para o número de caracteres que devem ser digitados, o que nos permitirá entrar dados sem a necessidade de pressionar a tecla **<ENTER>**.

Acrescente esta linha à sub-rotina:

```
1050 IF LEN(S$)=MX THEN RETURN
```

A variável **MX**, que indica o tamanho máximo do campo, deve ser especificada antes de se chamar a rotina. Para testá-la, acrescente estas linhas ao programa principal:

```
20 PRINT "TAMANHO DO CAMPO ";
30 INPUT MX
```

Podemos ainda utilizar uma outra tecla — como **<SCAPE>** — para concluir a entrada de dados.

Substitua a linha 1025 por:

```
1025 IF C=TX THEN RETURN
```

Note que **TX** deve ser especificada pelo programa que chama a sub-rotina.

Acrescente estas linhas, para testar a nova opção:

```
40 PRINT "CODIGO DE TERMINO ";
50 INPUT TX
```

Uma rotina de entrada de dados também não estaria completa se não oferecesse ao usuário a possibilidade de corrigir erros, usando a tecla **<BACKSPACE>** ou <-. Adicione ao programa as seguintes linhas:

```
1026 IF C=8 AND LEN(S$)>0 THEN
1070
1070 LET S$=LEFT$(S$,LEN(S$)-1)
1080 GOTO 1040
```

```
1026 IF C=8 AND LEN(S$)>0 THEN
1070
1070 LET S$=S$(TO LEN(S$)-1)
1080 GOTO 1040
```

```
1026 IF C=114 AND LEN(S$)>0
THEN 1070
1070 LET S$=S$(TO LEN(S$)-1)
1080 GOTO 1040
```

A linha 1026 verifica se a variável de saída já tem no mínimo um caractere e se a tecla de correção foi pressionada (código ASCII 8; em se tratando do ZX-81, deve ser pressionada a tecla <-, cujo código é 114).

Em caso positivo, a rotina desvia para a linha 1070, que extrai o caractere apagado e vai para a linha 1040. Esta faz retornar o cursor, imprimindo-a sobre a posição anterior.

ENTRADAS SECRETAS

Se quiser que os caracteres digitados não apareçam na tela — para a entrada de uma senha secreta, por exemplo —, substitua a linha 1040 por:

```
1040 IF SX=1 THEN PRINT C$;
```

A variável **SX** indica se o caractere digitado deve ser mostrado (**SX** = 1) ou não (**SX** < >1). Para testar essa modificação, acrescente as próximas linhas ao programa principal:

```
60 PRINT "ENTRADA INVISIVEL
(SIM=0) ";
70 INPUT SX
```

ENTRADAS NUMÉRICAS

Podemos tornar nossa rotina de entrada ainda mais versátil. Uma opção interessante consiste em utilizar a chamada *censura de entrada*, por intermédio da qual fazemos com que a rotina ignore automaticamente toda tecla que não esteja incluída dentro de um determinado conjunto de entrada. Essa alternativa nos permite estabelecer que todas as teclas sejam maiúsculas, ou que não ocorram certos caracteres, como vírgulas, ou, também, que a entrada seja apenas numérica.



O que faz a instrução LINE INPUT?

Esta resposta é dirigida aos usuários de computadores TRS-80 (com BASIC nível II), ZX-81, Sinclair, Apple e TK-2000. O BASIC padrão de suas máquinas não dispõe da instrução **LINE INPUT**, mas a sub-rotina apresentada neste artigo pode substituí-la muito bem.

Como o nome sugere, a instrução **LINE INPUT** permite a entrada de uma linha de texto pelo teclado. Essa linha é armazenada em uma variável literal ou *string*, nomeada pela instrução. Por exemplo:

```
LINE INPUT X$
```

Não podemos colocar outras variáveis para entrada pela mesma instrução.

Ao contrário da instrução **INPUT**, **LINE INPUT** não imprime na tela o ponto de interrogação — sinal de prontidão para o usuário —, o que dá uma maior liberdade de programação.

O **LINE INPUT** aceita uma linha de texto contendo vírgulas, pontos e vírgulas e outras pontuações — que nos obrigariam ao uso de aspas, para delimitar o string de entrada, caso utilizássemos uma instrução **INPUT**. Assim, é ideal para a entrada em processamento de texto

O procedimento é, na verdade, bastante simples. Para obter uma rotina de entrada numérica, por exemplo, basta que acrescentemos:

```
1027 IF NOT (C>=43 AND C<=46)
AND NOT (C>=48 AND C<=57) THEN
GOTO 1020
```

```
1027 IF C<>21 AND C<>22 AND
C<>26 AND C< >27 AND NOT (C>=28
AND C<=37) THEN GOTO 1020
```

Essa linha verifica se o caractere entrado está entre 0 e 9 ou se é um sinal de mais, de menos, vírgula ou ponto. Se não for um desses, retorna para a linha 1020, de espera.

CONSULTA AOS ASTROS

- CARREIRA/DINHEIRO
- CONTATOS/VIAGENS
- FAMILIA
- AMOR
- PERSONALIDADE

Muitas pessoas não saem de casa sem consultar o horóscopo: Você pode não ser uma delas, mas certamente se divertirá ouvindo o que o computador tem a dizer sobre seu destino.

Astrologia e horóscopos intrigam as pessoas desde a Antiguidade, e têm sido motivo de muita controvérsia. Cientistas de histórico tão distinto, como Einstein e Jung, acreditavam na influência dos astros sobre as pessoas — hipótese rejeitada pela maioria dos seus contemporâneos. Estudos multidisciplinares recentes, contudo, têm apresentado algumas evidências difíceis de se refutar. Seja como for, o papel dos astros na determinação da personalidade e do destino dos seres humanos é uma questão que permanece aberta à discussão.

Ainda que o tema não o apaixone, você poderá passar horas muito agradáveis com seus amigos em torno de um computador astrólogo. Para atribuir a máquina esse papel, basta digitar o programa aqui apresentado. Ele fornecerá características de personalidade de cada um de acordo com seu signo solar, assim como as previsões para o próximo ano. Estas se dividem em quatro ca-



CAPRICÓRNIO



tegorias: Dinheiro/Carreira, Contatos/Viagens, Família e Amor.

Como os horóscopos publicados nos jornais, o nosso também não leva em conta a hora exata do nascimento — dado que os astrólogos usam para fornecer uma leitura mais completa do mapa astral de seus clientes.

OPÇÕES

O programa pede a data de nascimento do consulente e diz o seu signo. Oferece-lhe, em seguida, duas opções: obter um perfil de sua personalidade ou as previsões para o próximo ano.

Se o consulente escolher a primeira alternativa, o programa selecionará duas sentenças relacionadas ao seu signo, das oito que tem guardadas na memória. Se optar pela segunda, terá que indicar um dos seguintes tópicos: Dinheiro/Carreira, Contatos/Viagens, Família e Amor. Duas previsões vão então aparecer. A máquina tem somente duas guardadas para cada tópico (por signo); portanto, não haverá nenhuma variação na resposta, se você fizer novamente a mesma escolha.

O PROGRAMA

Uma parte do programa é específica para cada computador. As linhas **DATA** são comuns a todos, embora algumas alterações sejam necessárias, como mostramos mais adiante.

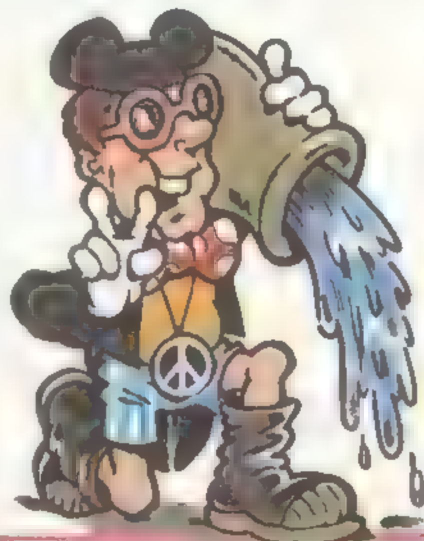


```
 5 POKE 23658,8
 10 INPUT "DIGITE SUA DATA DE
 NASCIMENTO   (DIA,MES,ANO) ";
 X,Y,Z
 20 IF X<1 OR X>31 OR Y<1 OR Y
 >12 THEN  GOTO 10
 40 GOSUB 700
 90 CLS : PRINT AT 1,1;"SEU SI
 GNO E "; INVERSE 1;A$
100 PRINT AT 5,1;" VOCE DESEJA
 :"''TAB 3;"<1> PERFIL DE PERSO
NALIDADE"'TAB 3;"<2> PREVISAO
PARA 1988"'TAB 3;"<N> ENTRAR N
OVA DATA"
120 LET K$=INKEY$: IF K$<"1"
OR K$>"5" AND K$<>"N" THEN
GOTO 120
130 IF K$="2" THEN  GOTO 300
140 IF K$="N" THEN  GOTO 10
150 CLS : PRINT AT 0,15-((LEN
(A$))/2);A$
160 FOR B=1 TO 2: LET A=INT (
RND*4)+1
170 FOR N=1 TO A: READ B$,C$,D
$: NEXT N
180 PRINT '': PRINT B$'C$'D$:
NEXT B: PAUSE 0: GOSUB 700:
GOTO 500
300 CLS : PRINT AT 8,8;"PREVIS
AO SOBRE :"''TAB 4;"<1> DINHEI
RO/CARREIRA"'TAB 4;"<2> CONTAT
OS/VIAGENS"'TAB 4;"<3> FAMILIA
 "'TAB 4;"<4> AMOR "
310 LET K$=INKEY$: IF K$=""
THEN  GOTO 310
320 IF K$<"1" OR K$>"4" THEN
GOTO 310
330 RESTORE (4000+320*(ST-1)):
FOR T=1 TO (VAL K$)-1: FOR N=0
TO 7: READ B$: NEXT N: NEXT T
335 CLS : PRINT AT 1,15-((LEN
A$)/2);A$'''
340 FOR T=0 TO 7: READ B$:
PRINT B$': NEXT T
500 PRINT AT 21,10;"OUTRA VEZ
?"
510 IF INKEY$="S" THEN  GOTO
90
520 IF INKEY$<>"N" THEN  GOTO
510
530 STOP
700 FOR T=1 TO 12: RESTORE (
1000+250*(T-1))
710 READ A$,A,B,C,D
720 IF B=Y AND X>=A THEN  LET
ST=T: LET T=12: NEXT T: RETURN

730 IF D=Y AND X<=C THEN  LET
ST=T: LET T=12: NEXT T: RETURN

740 NEXT T: RETURN
```

AQUÁRIO





```
10 PCLEAR 1
20 DIM H$(11,7,3),D(3,11),S$(11
),P$(11,3,7)
30 FOR K=0 TO 11:READ S$(K),D(0
,K),D(1,K),D(2,K),D(3,K):FOR J=
0 TO 7:FOR L=0 TO 2:READ H$(K,J
,L):NEXT L,J,K
40 FOR K=0 TO 11:FOR J=0 TO 3:F
OR L=0 TO 7:READ P$(K,J,L):NEXT
 L,J,K
50 CLS:INPUT"DIGITE SUA DATA DE
 NASCIMENTO   (DIA,MES E ANO)";
D,M,A:D=INT(D):M=INT(M)
60 IF D<1 OR D>31 OR M<1 OR M>1
2 OR (D>30 AND(M=4 OR M=6 OR M=
9 OR M=10)) OR (D>29 AND M=2)TH
EN 50
70 SS=0
80 IF M<D(1,SS)+12*(SS=2 AND M=
1) OR M>D(3,SS)-12*(SS=2 AND M=
12) OR (M=D(1,SS) AND D<D(0,SS)
) OR(M=D(3,SS)AND D>D(2,SS)) TH
EN SS=SS+1:GOTO 80
90 CLS:PRINT"SEU SIGNO E ";S$(S
S)
100 PRINT:PRINT TAB(10)"VOCE DE
SEJA:"
110 PRINT:PRINT TAB(3)"1- PERFI
L DE PERSONALIDADE"
120 PRINT TAB(3)"2- PREVISAO PA
RA 1988":PRINT:PRINT TAB(3)"N-
ENTRAR NOVA DATA"
130 K$=INKEY$:IF K$<"1" OR(K$>"
2" AND K$<>"N")THEN 130
140 IF K$="2" THEN 190 ELSE IF
K$="N" THEN 50
150 CLS:PRINT @16-LEN(S$(SS))/2
,S$(SS):PRINT
```

PEIXES



```
160 C=8·FOR K=1 TO 2·B=RND(8) 1
·IF B<>C THEN C=B ELSE K=1:NEXT
170 FOR L=0 TO 3·PRINT @L*32+K*
128,HS(SS,C,L)·NEXT L,K
180 GOTO 240
190 CLS·PRINT @2,"VOCE QUER SAB
ER SOBRE:"
200 PRINT·PRINT"  1- DINHEIRO/C
ARREIRA"·PRINT"  2- CONTATOS/VI
AGENS":PRINT"  3- FAMILIA":PRIN
T"  4- AMOR"
210 KS=INKEYS·IF KS<"1" OR KS>"
4" THEN 210
220 CLS·PRINT @16-LEN(SS(SS))/2
,SS(SS):PRINT
230 FOR K=0 TO 7:PRINT @64+32*K
-32*(K>3),PS(SS,VAL(KS)-1,K):NE
XT
240 PRINT @448," OUTRA VEZ (S/N
) ?"
250 KS=INKEYS:IF KS<>"S" AND KS
<>"N" THEN 250
260 IF KS="S" THEN 90 ELSE CLS:
END
```



```
20 DIM HS(11,7,3),D(3,11),SS(11
),PS(11,3,7)
30 FOR K=0 TO 11:READ SS(K),D(0
,K),D(1,K),D(2,K),D(3,K):FOR J=
0 TO 7·FOR L=0 TO 2:READ HS(K,J
,L)·NEXT L,J,K
40 FOR K=0 TO 11:FOR J=0 TO 3·F
OR L=0 TO 7· READ PS(K,J,L):NEX
T L,J,K
50 CLS:INPUT"DIGITE SUA DATA DE
 NASCIMENTO (D,M,A) ";D,M,A·D=I
NT(D):M=INT(M)
60 IF D<1 OR D>31 OR M<1 OR M>1
2 OR (D>30 AND (M=4 OR M=6 OR M
=9 OR M=11)) OR (D>29 AND M=2)
THEN 50
70 SS=0
80 IF M<D(1,SS)+12*(SS=2 AND M
1) OR M>D(3,22)-12*(SS=2 AND M=
12) OR (M=D(1,SS) AND D<D(0,SS)
) OR (M=D(3,SS) AND D>D(2,SS))
THEN SS=SS+1  GOTO 80
90 CLS·PRINT "SEU SIGNO E ":SS(
SS)
100 PRINT·PRINT TAB(10);"VOCE D
ESEJA ·"
110 PRINT:PRINT TAB(3);"1- PERF
IL DE PERSONALIDADE"
120 PRINT TAB(3);"2- PREVISAO P
ARA 1988"·PRINT·PRINT TAB(3),"N
- ENTRAR NOVA DATA"
130 KS=INKEYS IF KS<"1" OR (KS>
"2" AND KS<>"N") THEN 130
140 IF KS="2" THEN 190 ELSE IF
KS="N" THEN 50
150 CLS· PRINT TAB(20 LEN(SS(SS
))/2):SS(SS)·PRINT
160 C=8· FOR K=1 TO 2·B=8*RND(-
TIME)-1:IF B<>C THEN C=B ELSE K
=1·NEXT
170 FOR L=0 TO 3·LOCATE((L*32+K
*128)MOD32)+4,INT((L*32+K*128)/
32):PRINT HS(SS,C,L):NEXT L,K
180 GOTO 240
190 CLS PRINT TAB(10),"PREVISAO
 SOBRE "
200 PRINT PRINT" 1  DINHEIRO/CA
RREIRA":PRINT" 2- CONTATOS/VIAG
ENS" PRINT" 3- FAMILIA"·PRINT"
4- AMOR"
210 KS=INKEYS·IF KS<"1" OR KS>"
4" THEN 210
220 CLS PRINT TAB(20-LEN(SS(SS)
)/2);SS(SS):PRINT
230 FOR K=0 TO 7:LOCATE(64+32*K
 32*(K>3),MOD32,INT((64+32*K-32
*(K>3))/32) PRINT PS,SS,VAL(KS)
 1,K)·NEXT
240 PRINT TAB(10),"OUTRA VEZ (S
/N) ?"
250 KS=INKEYS IF KS<>"S" AND KS
<>"N" THEN 250
260 IF KS "S" THEN 90 ELSE CLS
END
```



```
10  HOME
20  DIM HS 11 7 3 ,D(3,11,,SS(1
1),PS(11,3,7)
30  FOR K = 0 TO 11  READ SS(K)
,D(0,K),D(1,K,,D 2,K,,D(3,K)  F
OR J = 0 TO 7  FOR L = 0 TO 2
READ HS(K,J,L)  NEXT L,J,K
40  FOR K  0 TO 11  FOR J = 0
TO 3  FOR L - 0 TO   READ PS(K
,J,L): NEXT L,J,K
```

```
50  HOME   PR.NT "DIGITE SUA DA
TA DE NASCIMENTO (D,M,A) ": INP
UT ..M,A D     IN   D) M =  INT
 M,
6,  IF D < . OR D > ". OR M < 1
  R M > .? OR  . > .( AND (M -
4 OR M - 6  R M   9 OR M = .1))
 R (D >   ) AND M = 2  THEN 50
7(  .
9(  IF M < L  .,SS)   1. * (SS
 . ANL M = .) OR M > D.(.SS) +
.2 * (SS   2 AND M   .2) OR (M
   . SS) ANI D < ( (0,SS)) OR (
M   D . SS) AND D > D(.,SS)) TH
EN SS = SS + 1: GOTO 80
9)  H.ME   PRINT "SEU SIGNO E "
.S.SS
.00  PRIN    HTAB (10)  PRINT "
V   E DESEJA "
 10  PRINT   PR NT  TAB( 3)"1
PERFIL DE PERSONALIDADE"
..0  PRINT  TAB( 3 "2  PREVISAO
 PARA 1988": PRINT : PRINT  TAB
  ( 'N  ENTRAR NOVA DATA"
 30   ET K"  IF KS < "1" OR (KS
 . ".  ANL K" <  > "N") THEN 13
C
 4   IF KS    "2" THEN .90
145  IF KS = "N" THEN 50
.50  H ME   LET CE   20 -  LEN
 SS SS)) / 2  HTAB (CE)  PRINT
S$(SS): PRINT
160 C    8  FOR K   1 TO . B
INT ( RND .1) * 8)  IF B <  > C
 THEN C = B  IF B = C THEN K
1  NEXT
170  FOR L = 0 TO 3  LET WW = L
 * 32 + K * 128: HTAB (WW - 32
*  INT (WW / 32))  VTAB ( INT (
WW / 32)): PRINT .IS(SS.C,L)  NE
XT L,K
180  GOTO 240
190  HOME : PRINT  TAB( 10)"VOC
E DESEJA:"
200  PRINT : PRINT "  1- DINHEI
RO/CARREIRA": PRINT "  2- CONTA
TOS/VIAGENS": PRINT "  3- FAMIL
IA": PRINT "  4- AMOR"
210  GET K$: IF K$ < "1" OR K$
> "4" THEN 210
220  HOME :CE = 20 -  LEN (S$(S
S)) / 2: HTAB (CE): PRINT S$(SS
): PRINT
230  FOR K = 0 TO 7: LET YY = 6
4 + 32 * K + 32 * (K > 3): HTAB
 (YY - 32 *  INT (YY / 32)): VT
AB ( INT (WW / 32)): PRINT P$(S
S. VAL (K$) - 1,K): NEXT
240  VTAB (18): PRINT " OUTRA V
EZ (S/N) ?"
250  GET K$  IF K$ <  > "S" AND
 K$ <  > "N" THEN 250
260  IF K$ = "S" THEN 90
270  IF K$ = "N" THEN  HOME : E
ND
```

ÁRIES

LINHA DATA

As linhas a seguir podem ser introduzidas como estão no MSX e no Apple. Se você possui um TRS-Color, deve usar aspas em todas as linhas **DATA** que contêm dois pontos (:) ou vírgula. Os usuários do Spectrum precisarão usar aspas em todas as linhas, exceto naquelas que contêm números.

```
1000 DATA PEIXES.20.2,20 3
.010 DATA Você é o méd.um do
1020 DATA "zodiaco, sendo o mais intuitivo"
1030 DATA e sensit.vo dos signos
.040 DATA voc. é piedoso e
1050 DATA se envo.ve prontamente com os
.060 DATA problemas de outras pessoas
1070 DATA "Você absorve as atmosferas, e pr
ec.sa de
.080 DATA "um amb.ente alegre  envo.vendo-s
e com
 090 DATA aqueles que o fazem se sentir tra
nq.  .o
110  DATA "você gosta de álcool  e precisa
ser"
...0 DATA cu.dadoso para não se embebedar
1120 DATA " "
.. 30 DATA Você é tão generoso que
1140 DATA frequentemente não tem dinheiro
. 50 DATA o bastante para você mesmo
1 60 DATA Você é um grande artista  Tem dom
..70 DATA "para mús..a, dança  desenho"
..80 DATA ou fotografia
..90 DATA "você vive e respira religião,"
1200 DATA mas não necessar.amente no
```

TOURO

```
.210 DATA sent.do forma. da pa.avra
1220 DATA "Compat.bil dade  V.rgem  Gêmeos
"
1230 DATA Sag tár.o
.240 DATA "Cores  verde  azul "
1250 DATA AQUARIO,2... 19 2
1260 DATA Você cons.dera seus am.gos como
1270 DATA "sendo parte de sua famí.ia  tant
o"
1280 DATA quanto seus parentes
1.90 DATA você é .ndependente desde
.300 DATA "pequeno, mas terá uma casa"
1310 DATA sempre cheia de amigos
1320 DATA Você frequenta c.ubes e assoc.açõ
es
1330 DATA mais do que os outros signos
1340 DATA e pode ser um membro do CND
1350 DATA "Você é .nvent.vo e inovador,"
1360 DATA e tem idéias e concepções
1370 DATA avançadas
.380 DATA Você tem dificuldade em se
1390 DATA "reportar a alguém  e pode ser"
.400 DATA .so.ado por causa disso
1410 DATA "Você veste roupas bizarras."
1420 DATA que são mais interessantes
.430 DATA do que elegantes
1440 DATA "Você esco.he ciênc.as sociais,"
1450 DATA politica ou engenharia eletrônica
.460 DATA como carreira
1470 DATA "Compatibilidade  Capricórnio, L.
bra "
```

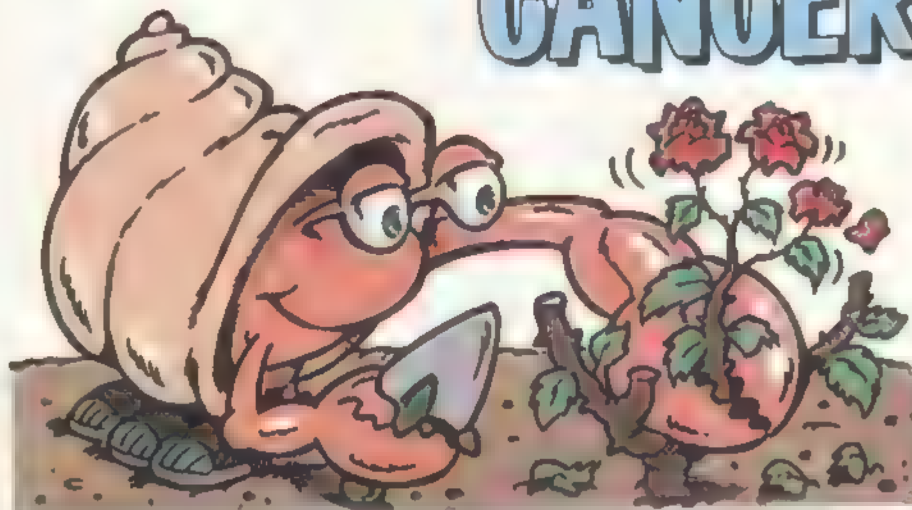


```
1480 DATA "Cor  azul ferrete "
1490 DATA " "
1500 DATA CAPRICORNIO,22,12,20,1
1510 DATA Você é o mais ambicioso dos
1520 DATA "signos  traçando planos de"
1530 DATA carreira desde pequeno
1540 DATA As crianças de Capricórnio
1550 DATA parecem ser adultos sérios e
1560 DATA maduros desde o nascimento
1570 DATA Você precisa se resguardar
1580 DATA contra o reumatismo e todos os
1590 DATA alimentos que atacam a espinha
1600 DATA As carreiras para o capricorniano
1610 DATA "se relacionam a matemática,"
1620 DATA "contabilidade ou finanças "
1630 DATA "Você é econômico, mas"
1640 DATA tudo o que compra é de
1650 DATA qualidade e feito para durar
1660 DATA Você é caracterizado pela sua
1670 DATA conduta irrepreensível e olhar
1680 DATA sério
1690 DATA Você tem mais conhecidos do que
1700 DATA "amigos  seus verdadeiros"
1710 DATA amigos são sua família
1720 DATA "Compatibilidade  Touro, Câncer "
1730 DATA Aquario
1740 DATA "Cores  Preto e tons escuros "
1750 DATA SAGITARIO,23,11,21,12
1760 DATA Você é um otimista de nascimento
1770 DATA e é divertido e com um grande
1780 DATA senso de humor
1790 DATA Você gosta de festas e de
1800 DATA "participar de jogos, mas"
1810 DATA seu fraco são belas coxas
1820 DATA Você adora viajar e conhecer
1830 DATA novas pessoas
1840 DATA " "
1850 DATA "Você tem bom ouvido para línguas
 "
1860 DATA e pode falar falar fluentemente
1870 DATA muitas delas
1880 DATA "Carreira  jornalista, radialista
,"
1890 DATA "professor  filósofo "
1900 DATA esportista
1910 DATA "Você é muito dinâmico, e"
1920 DATA acha difícil ficar parado em
1930 DATA um mesmo lugar o tempo bastante
1940 DATA para ser monógamo
1950 DATA "Você adora animais, e o jogo,"
1960 DATA "infelizmente  pode ser uma fraqu
eza "
1970 DATA "Compatibilidade  Peixes, Virgem,
"
1980 DATA Gêmeos
1990 DATA "Cores  azul royal, violeta"
2000 DATA ESCORPIAO,24,10,22,11
2010 DATA "Você é o mais apaixonado dos sig
nos,"
2020 DATA "emotivo, e, se contrariado,"
2030 DATA pode ser vingativo
2040 DATA Você é reconhecido por seus
2050 DATA olhos penetrantes e por seu
2060 DATA nariz xereta
2070 DATA Seis meses antes ou depois do
2080 DATA nascimento de um Escorpião
2090 DATA um parente falece
2100 DATA Você é interessado em aprender
2110 DATA e adora resolver mistérios
2120 DATA " "
2130 DATA Homens e mulheres de Escorpião sã
,
2140 DATA m[illegible] atra[illegible]
2150 DATA " "
2160 DATA [illegible]
2170 DATA "[illegible] da [illegible]"
2180 DATA [illegible] poder
2190 DATA [illegible]
2200 DATA mas tud[illegible] que [illegible]
2210 DATA "[illegible]"
2220 DATA "Compatibilidade [illegible]"
2230 DATA "C[illegible] Marrom[illegible]"
2240 DATA "
2250 DATA [illegible]BRA,[illegible]
2260 DATA [illegible] divertid[illegible]
2270 DATA "[illegible]"
2280 DATA suas [illegible]
2290 DATA "Você [illegible]
2300 DATA Pre[illegible]sa [illegible] de
2310 DATA outras pessoas
2320 DATA "Você é um excelente [illegible]
2330 DATA e é muito háb[illegible] para [illegible]
2340 DATA atmosfera de harmonia
```

GÊMEOS



CÂNCER



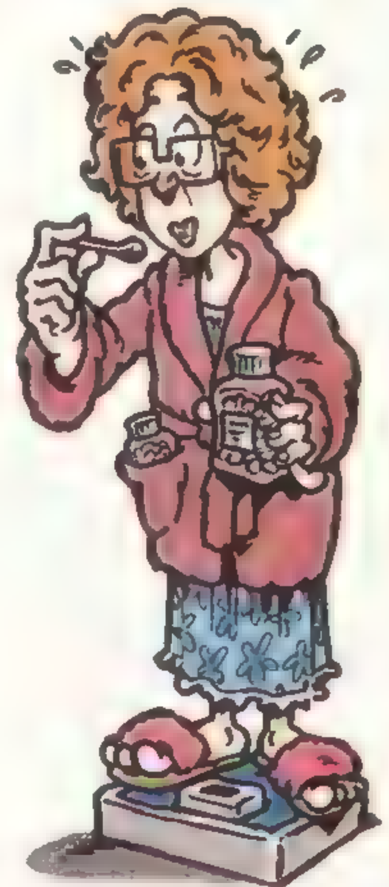
```
2350 DATA "Você tem [illegible]vinhas nas bo[illegible] das
"
2360 DATA e [illegible]abina bem [illegible] E [illegible]
2370 DATA das [illegible]
2380 DATA você às vezes f[illegible] a [illegible]
2390 DATA e [illegible] fica muit[illegible] tempo [illegible]
2400 DATA As carreiras para [illegible]
2410 DATA "[illegible] ale[illegible] "
2420 DATA "esteticista [illegible]
2430 DATA "advogado "
2440 DATA Seu b[illegible] senso para [illegible]
2450 DATA da harm[illegible] as suas [illegible]
2460 DATA pela boa esc[illegible]lha das [illegible]
2470 DATA "Compatibilidade Aquár[illegible] Ar[illegible]
"
2480 DATA Escorpiã
24[illegible] DATA "[illegible]res de tom pastel "
25[illegible] DATA VIRGEM [illegible]
[illegible] DATA [illegible]
[illegible] DATA a [illegible]dar nenhum sem uma [illegible]
[illegible] DATA de [illegible] para [illegible]
[illegible] DATA "[illegible] tem um [illegible] sa dáve[illegible]
[illegible] DATA [illegible]
[illegible] DATA aparên[illegible]a
[illegible] DATA "Você é [illegible]"
[illegible] DATA será semp[illegible] [illegible]mente [illegible]
[illegible] DATA " "
2600 DATA [illegible]
26[illegible] DATA [illegible] mu[illegible]ta atenção aos
2620 DATA [illegible] em [illegible] aspectos da
vida
26[illegible] DATA "[illegible]arreiras [illegible]tecár[illegible] "
264[illegible] DATA "estat[illegible] cientista"
26[illegible] DATA [illegible] méd[illegible]
266[illegible] DATA [illegible] é [illegible]mente [illegible]
26[illegible] DATA "[illegible] fam[illegible] é muito [illegible]ande "
2680 DATA desordem deixa você [illegible]
2690 DATA "Você é [illegible] por nat[illegible]reza "
2700 DATA "e [illegible] se trabalhar em um escritório
"
2710 DATA será muito organizada
2720 DATA "Compatibilidade Peixes, Gemeos,
"
2730 DATA Sagitário
2740 DATA "Cor Azul escuro "
2750 DATA LEAO 24 7,23,8
2760 DATA Ambos os sexos vestem as
2770 DATA roupas e jóias mais caras e
2780 DATA são incrivelmente vaidosas
2790 DATA "Como os leões, você pode"
2800 DATA ter uma grande cadeleira
2810 DATA dourada
2820 DATA "Você adora drama e gosta de"
2830 DATA passar seu tempo livre no
2840 DATA teatro ou cinema
2850 DATA Seu gosto por comidas fortes
2860 DATA é um perigo para seu coração
2870 DATA Você é propenso a ataques cardiac
os
2880 DATA Você é generoso com seu dinheiro
2890 DATA - mas não deixa de ser
2900 DATA prevenido
2910 DATA"Carreiras joalheiro, ator "
2920 DATA "professor, líder em alguma funçã
o "
2930 DATA " "
2940 DATA Seu lar pode ser uma grande
2950 DATA casa desunida vizinhos são
2960 DATA ameaça ao seu espaço
2970 DATA "Compatibilidade Cancer Aries "
2980 DATA "Cores Ouro e Ambar "
2990 DATA " "
3000 DATA CANCER,22.6 23.7
3010 DATA "Você é muito tímido, escondendo"
3020 DATA seus lindos olhos
3030 DATA sob pálpebras submissas
3040 DATA Você sempre permanecerá preso a
3050 DATA seus parentes(principalmente
3060 DATA sua mãe) mesmo depois de casado
3070 DATA "Você é normalmente tímido, mas"
```

LEÃO



```
3080 DATA lutaria até a morte para
3090 DATA proteger sua família
3100 DATA Sua casa é seu castelo e somente
3110 DATA amigos muito especiais
3120 DATA a frequentam
3130 DATA Você gostaria de viver perto
3140 DATA "da água - um rio, riacho ou"
3150 DATA à beira-mar
3160 DATA "Carreiras  enfermeiro  jardineiro  marinheiro."
3170 DATA "cozinheiro, professor ou fotógrafo "
3180 DATA " "
3190 DATA "Você se prende ao passado, e"
3200 DATA adora colecionar antiguidades
3210 DATA Você tem boa memória fotográfica
3220 DATA "Compatibilidade  Leão, Escorpião "
3230 DATA "Cores  Branco e prata "
3240 DATA " "
3250 DATA GEMEOS,22,5,21,6
3260 DATA "Você é do tipo esbelto, e tem"
3270 DATA "membros longos e delgados "
3280 DATA " "
3290 DATA "Você fala pelos cotovelos, e"
3300 DATA poderia passar um dia inteiro
3310 DATA falando somente banalidades
3320 DATA Você é o mais comunicativo do
3330 DATA "zodíaco, reconhecido pelo"
3340 DATA modo como gosta de falar com as mãos
3350 DATA Ambos os sexos sofrem dos nervos
3360 DATA "e insônia, e devem se precaver"
3370 DATA contra a estafa
3380 DATA "Sua casa será sempre deserta."
3390 DATA você estará sempre visitando um
3400 DATA vizinho ou amigo
3410 DATA Você se sente sufocado num
3420 DATA "relacionamento mais duradouro "
3430 DATA "e evita se prender a um companhe[illegible]a "
3440 DATA "Carreiras  [illegible]sta  vendedor, "
3450 DATA "professor, escritor "
3460 DATA " "
3470 DATA "Compatibilidade  Sagitário,"
3480 DATA "[illegible] Virgem "
3490 DATA "Cor  amarelo "
3500 DATA ARIES 2[illegible] 20 [illegible]
3510 DATA "você adora gastar dinheiro, "
3520 DATA mas coloca suas necessidades em primeiro lugar
3530 DATA " "
3540 DATA "Você reage bem nas adversidades. "
3550 DATA "tem gênio forte  mas sabe"
3560 DATA "perdoar com facilidade "
3570 DATA "Você é fisicamente ativo, e"
3580 DATA "prefere esportes que possa praticar"
3590 DATA individualmente
3600 DATA "Você sofre de dores de cabeça, e "
3610 DATA "é propensa a sofrer cortes ou contusões "
3620 DATA " "
3630 DATA Você aprecia muito a companhia
3640 DATA "de outras pessoas, especialmente "
3650 DATA daquelas que não interferem no seu modo de ser
3660 DATA "Carreiras  açougueiro,"
3670 DATA esportista
3680 DATA " "
3690 DATA "Suas roupas são ousadas, "
3700 DATA "sempre justas, mas de desenhos"
3710 DATA simples
3720 DATA "Compatibilidade  Touro, Libra, Leão "
3730 DATA "Cor  Vermelho "
3740 DATA " "
3750 DATA TOURO,21 4,2, 5
3760 DATA Você é afetuoso e
3770 DATA romântico
3780 DATA " "
3790 DATA "você é cuidadoso com dinheiro, "
3800 DATA precisa de segurança financeira mais
3810 DATA do que qualquer outro signo
3820 DATA "Embora seja normalmente calmo, "
3830 DATA "você pode se tornar temível, "
3840 DATA "especialmente se for para defender seu cônjuge "
3850 DATA Seu temperamento de apreciar as
3860 DATA boas coisas frequentemente o
3870 DATA leva a um médico para a indicação de uma dieta
3880 DATA "Você pode se tornar muito possessivo"
3890 DATA "no amor, e espera que seu parceiro"
3900 DATA rejeite o sexo oposto
3910 DATA "Você é muito sensual, gosta do"
3920 DATA contato de tecidos sedosos
3930 DATA contra a pele
3940 DATA "Carreiras  arqueologista, banqueiro,"
3950 DATA "fazendeiro, empresário de"
3960 DATA vendas
3970 DATA "Compatibilidade  Aries, Capricórnio "
3980 DATA "Cores  Azul, rosa "
3990 DATA " "
4000 DATA Em fevereiro, seu trabalho será
4010 DATA "mais isolado do que o normal, "
4020 DATA e você terá que impor muita
4030 DATA auto disciplina
4040 DATA "Na primavera, terá"
4050 DATA despesas extras com
4060 DATA viagens
4070 DATA " "
4080 DATA Você vai aproveitar um longo
4090 DATA "feriado em Outubro/Novembro"
4100 DATA aprofundando-se nos mistérios
4110 DATA dos antigos impérios
4120 DATA Em junho, você deverá
4130 DATA adotar atitudes mais sérias
4140 *DATA "diante dos seus problemas  desde"
4150 DATA os cotidianos até os mais graves
4160 DATA Em junho você trabalhará duro
4170 DATA e gastará dinheiro com sua
4180 DATA "casa, provavelmente comprando"
4190 DATA eletrodomésticos
4200 DATA Em Julho/Agosto você enfrentará
4210 DATA um dilema entre as prioridades
4220 DATA do trabalho e da sua família
4230 DATA " "
4240 DATA Uma amizade que começará
4250 DATA em janeiro pode se mostrar duradouro.
4260 DATA " "
4270 DATA " "
4280 *DATA "Em Outubro, você enfrentará um"
4290 DATA "período de mau humor, envolvido"
4300 DATA por uma amizade passageira que
4310 DATA o deixará mal com seus amigos.
4320 DATA Por muitos anos você terá a
4330 DATA vantagem de ter um planeta
4340 DATA poderoso regendo sua
4350 DATA carreira
```

VIRGEM



```
4360 DATA Nos últimos dois anos você
4370 DATA tem estado angustiado por ter
4380 DATA ou não escolhido o caminho certo.
4390 DATA Em 1988 você terá a resposta
4400 DATA Entre Fevereiro e Abril você
4410 DATA viajará curtas distâncias para
4420 DATA encontrar seus entes queridos.
4430 DATA " "
4440 DATA Você provavelmente viajará
4450 DATA "no final do ano (Setembro/"
4460 DATA "Outubro), com amigos."
4470 DATA " "
4480 DATA Você vai passar o mês de Julho
4490 DATA "em casa, e deve tomar"
4500 DATA cuidado para não engordar
4510 DATA muito
4520 DATA Um desentendimento com
4530 DATA alguém da família pode
4540 DATA ocorrer em Março
4550 DATA " "
4560 DATA Um intenso relacionamento pode
4570 DATA "começar em setembro, começando"
4580 DATA de algo que para os outros
4590 DATA não passava de uma paquera.
```

LIBRA



```
4600 DATA Os homens podem se ver
4610 DATA comprometidos com a figura da
4620 DATA "mãe em Dezembro, mas é improváve
l"
4630 DATA que esta situação dure até Janeir
o.
4640 DATA No final de 1988 você pode
4650 DATA achar que conquistou uma posição
4660 DATA "de autoridade, com opiniões"
4670 DATA "claras e objetivas."
4680 DATA Você não deve ser muito
4690 DATA simpático com aquelas que não
4700 DATA "o são, deve pensar um pouco"
4710 DATA mais em você mesmo.
4720 DATA Você se tornará muito popular
4730 DATA "no trabalho em janeiro, graças"
4740 DATA a uma grande capacidade de comuni
cação.
4750 DATA " "
4760 DATA É provável uma grande viagem
4770 DATA ao exterior em Setembro. Tome
4780 DATA cuidado com roubos
4790 DATA " "
4800 DATA "Sua casa, em março, parecerá"
4810 DATA um campo de batalha. Tente
4820 DATA "evitar afirmações ambíguas, "
4830 DATA "que podem ocasionar um rompiment
o."
4840 DATA Você passará o mês de Abril
4850 DATA preocupado com a decoração de
4860 DATA sua casa.
4870 DATA " "
4880 DATA "Pode surgir um caso em Agosto,"
4890 DATA mas você deve tomar cuidado para
4900 DATA não ser nem muito dominante
4910 DATA nem exageradamente atencioso
4920 DATA Um relacionamento duradouro
4930 DATA surgirá com alguém que compartilh
a
4940 DATA alguma atividade ou interesse
4950 DATA "com você, em Novembro "
4960 DATA A posição dos planetas em Setembr
o
4970 DATA pode significar alguma mudança
4980 DATA "na sua carreira, que você planej
ava"
4990 DATA desde o ano passado.
5000 DATA Você estará financeiramente bem
5010 DATA "em 1988, e fará donativos a"
5020 DATA instituições de caridade que
5030 DATA cuidam de idosos.
5040 DATA Você estará envolvido em um grand
e
5050 DATA "negócio durante todo o ano."
5060 DATA a partir de fevereiro
5070 DATA " "
5080 DATA "Você se ligará a tantas associaç
ões,"
5090 DATA "particularmente de ensino superi
or"
5100 DATA que talvez esqueça de que
5110 DATA tem uma casa
5120 DATA Você passará por um período
5130 DATA "muito romântico em seu lar,"
5140 DATA em Janeiro.
5150 DATA " "
5160 DATA A partir de fevereiro começará
5170 DATA um ano de muita atividade para
5180 DATA "você, fora de casa "
5190 DATA " "
5200 DATA Um romance pode começar no
5210 DATA "Natal, mas talvez sem "
5220 DATA atração física
5230 DATA " "
5240 DATA "Uma relação mais profunda."
5250 DATA "até certo ponto frívola, pode"
5260 DATA "começar em Junho Esta pode"
5270 DATA ser uma relação ardente e dramáti
ca.
5280 DATA A despeito de algumas
5290 DATA "divergências, você trabalhará"
5300 DATA "duro, como fez ano passado, para
"
5310 DATA consolidar sua posição no trabalh
o
5320 DATA Você deve se precaver contra
5330 DATA "gastos desnecessários, tanto"
5340 DATA "seus como das outras pessoas da
casa."
5350 DATA "que tendem a ser extravagantes "
5360 DATA Você se envolverá com trabalho
5370 DATA "de caridade nos próximos anos "
5380 DATA sendo bom em aconselhar amigos
5390 DATA " "
5400 DATA Você vai querer viajar em Julho
5410 DATA mas terá que esperar por Agosto
5420 DATA até ter dinheiro o suficiente
5430 DATA " "
5440 DATA Você vai sentir que sua família
5450 DATA em peso quer que você faça
```

ESCORPIÃO



```
5460 DATA curso superior
5470 DATA " "
5480 DATA É possível que sua família não
5490 DATA receba bem um amigo seu que
5500 DATA é conhecido por ser muito otimist
a
5510 DATA " "
5520 DATA Da segunda semana de Novembro
5530 DATA até Dezembro existe a possibilida
de
5540 DATA de começar uma intensa relação
5550 DATA com uma pessoa mais velha
5560 DATA Um romântico caso de amor pode
5570 DATA "começar no início do ano, mas"
5580 DATA talvez desmorone se você não
5590 DATA deixar de pensar tanto no trabalh
o
5600 DATA Da segunda metade de Junho até
5610 DATA mês de agosto você estará
5620 DATA preocupado em afirmar sua carreir
a
5630 DATA " "
5640 DATA Financeiramente você estará
5650 DATA consolidando os ganhos que
5660 DATA "obteve em 1987, fruto de"
5670 DATA longo e cuidadoso planejamento.
5680 DATA Você vai se notar cada vez
5690 DATA mais interessado em discutir
5700 DATA ou falar sobre temas
5710 DATA políticos
5720 DATA Uma viagem ao exterior é
5730 DATA provável em Maio, provavelmente
5740 DATA cumprindo um circuito turístico
5750 DATA " "
```

SAGITÁRIO





```
5760 DATA Por um período de dez anos você
5770 DATA sentirá atração por mudar-se
5780 DATA para a qum lugar perto de água
5790 DATA " "
5800 DATA É provável que algum membro de
5810 DATA sua família se interesse por
5820 DATA uma religião em particular
5830 DATA ou algo similar
5840 DATA No final de Outubro um longo
5850 DATA caso de amor se iniciará.
5860 DATA " "
5870 DATA " "
5880 DATA Compromissos importantes serão
5890 DATA assumidos pelos homens no começo
5900 DATA "de Outubro, e pelas mulheres"
5910 DATA no começo de Novembro
5920 DATA Em junho você estará se
5930 DATA "realizando com sua carreira,"
5940 DATA fazendo muitas viagens
5950 DATA relacionadas a seu trabalho
5960 DATA Em Outubro você receberá
5970 DATA grande soma em dinheiro
5980 DATA fruto de antigos negócios.
5990 DATA " "
6000 DATA "Por todo este ano, como"
6010 DATA "foi o ano passado, você"
6020 DATA estará falando de sua carreira.
6030 DATA " "
6040 DATA Em Outubro/Novembro você será
6050 DATA uma companhia apenas agradável
6060 DATA e estará num estado
6070 DATA de muita despreocupação.
6080 DATA Em 1988 você vai hospedar um
6090 DATA "amigo recém-divorciado, ou"
6100 DATA um estudante em sua casa
6110 DATA " "
6120 DATA Um acontecimento inesperado no
6130 DATA verão 88/89 pode significar
6140 DATA uma mudança para nova casa
6150 DATA " "
6160 DATA Um romance nascerá em Setembro
6170 DATA desde que sua atitude
6180 DATA independente quanto a formar um
6190 DATA lar não estrague tudo.
6200 DATA Existe a possibilidade de
6210 DATA casamento ou início de alguma
6220 DATA relação permanente em
6230 DATA Outubro
6240 DATA O final de maio pode ser uma
6250 DATA boa época para você mudar de
6260 DATA "emprego, se acha que existe"
6270 DATA essa necessidade
6280 DATA Você pode achar qualquer troca
6290 DATA de emprego financeiramente
6300 DATA vantajosa - este pode ser o
6310 DATA final de uma era
6320 DATA Em Outubro/Novembro você
6330 DATA estará muito alegre e sociável
6340 DATA mas em Dezembro deve tomar cuidad
o
```

```
6350 DATA para não ser muito mandão.
6360 DATA Na primavera você vai estar
6370 DATA "em algum lugar muito quente."
6380 DATA e vai estar em muito
6390 DATA boa companhia
6400 DATA Você pode ter que cuidar de
6410 DATA alguma senhora idosa da família.
6420 DATA Mas a obrigação pode ser aliviada
6430 DATA em Outubro e Novembro
6440 DATA Você vai aproveitar muito o
6450 DATA "Natal, mas cuidado para não"
6460 DATA quebrar um braço nas festas
6470 DATA " "
6480 DATA 88 será um ano mais de amizades
6490 DATA do que de casos amorosos
6500 DATA " "
6510 DATA " "
6520 DATA Um romance de verão pode
6530 DATA "acontecer nas férias mas"
6540 DATA ambos os parceiros não
6550 DATA estarão apaixonados
6560 DATA Fevereiro e Abril serão meses
6570 DATA significantes para sua carreira
6580 DATA e você pode receber aumento
6590 DATA " "
6600 DATA Setembro será um mês muito
6610 DATA tranquilo financeiramente
6620 DATA você gastará bastante em
6630 DATA lazer
6640 DATA "Em setembro, uma disputa no"
6650 DATA trabalho pode o levar à
6660 DATA greve ou destacar sua
6670 DATA para a praça
6680 DATA Uma viagem ao exterior é
6690 DATA muito provável no começo
6700 DATA de Agosto
6710 DATA " "
6720 DATA Em junho você gozará a
6730 DATA companhia dos homens da
6740 DATA família e relaxará em casa
6750 DATA " "
6760 DATA Em Agosto você estará decorando
6770 DATA o jardim ou comprando
6780 DATA antiguidades para sua casa
6790 DATA " "
6800 DATA Em junho você pode iniciar um
6810 DATA relacionamento de pura atração
6820 DATA "física, sentimento de amor só"
6830 DATA depois de Outubro
6840 DATA "Em Outubro, você poderá"
6850 DATA ter uma criança
6860 DATA " "
6870 DATA " "
6880 DATA "Nos próximos treze anos, "
6890 DATA"você será capaz de influenciar,"
6900 DATA "sutilmente, as carreiras de"
6910 DATA muitas pessoas
6920 DATA Julho e Agosto serão meses muito
6930 DATA importantes para suas finanças
6940 DATA Suas aquisições em Agosto serão
6950 DATA compras inteligentes
6960 DATA Você deve tomar cuidado com o
6970 DATA "que fala em Julho/Agosto, para"
6980 DATA não se meter em apuros
6990 DATA " "
7000 DATA É provável que surja uma
7010 DATA grande viagem nas últimas
7020 DATA duas semanas de Dezembro e
7030 DATA no final do ano em geral
7040 DATA Em setembro você trará algum
7050 DATA estrangeiro ou forasteiro
7060 DATA para sua casa
7070 DATA " "
7080 DATA Você estará muito ativo em
7090 DATA "Setembro, e se dividirá"
7100 DATA entre seus compromissos sociais
7110 DATA e os afazeres domésticos
7120 DATA Você vai passar a maior
7130 DATA "parte de setembro em casa,"
7140 DATA "curtindo uma paixão, e vai"
7150 DATA causar estranheza a muitos
7160 DATA Os homens vão se sentir muito
7170 DATA "românticos no começo de Julho,"
7180 DATA e as mulheres durante Junho e
7190 DATA Julho
7200 DATA Você vai passar de Março a Maio
7210 DATA "pondo em ordem suas finanças,"
7220 DATA você se sente gastando muito
7230 DATA impulsivamente
7240 DATA Em dezembro você vai começar
7250 DATA a trabalhar ou mudar sua carreira
7260 DATA para alguma coisa que lhe dê
7270 DATA maior liberdade de comunicação
7280 DATA Aparecerão muitas oportunidades
7290 DATA "de viagem em 1988, especialmente
"
7300 DATA viagens aéreas
7310 DATA Uma viagem inesperada pode
7320 DATA "acontecer em Dezembro, e"
7330 DATA durante essa viagem pode
7340 DATA surgir um romance
7350 DATA " "
7360 DATA Em junho você pode se envolver
7370 DATA em tensas discussões na família
7380 DATA em torno de um membro idoso
7390 DATA ou inválido
7400 DATA Em agosto você vai gastar
7410 DATA bastante dinheiro em itens de
7420 DATA luxo para sua casa
7430 DATA " "
7440 DATA Um caso secreto que teve
7450 DATA seu início em Janeiro
7460 DATA deve acabar em
7470 DATA Fevereiro
7480 DATA Um romance pode começar no
7490 DATA "seu trabalho em Outubro, mas"
7500 DATA o parceiro vai acabar se
7510 DATA interessando por uma pessoa mais
velha
7520 DATA Depois de fevereiro você
7530 DATA vai se envolver com problemas
7540 DATA legais que afetam sua
7550 DATA carreira
7560 DATA Pode aparecer um ganho
7570 DATA "extra em Junho que"
7580 DATA você deve reinvestir
7590 DATA " "
7600 DATA Agosto vai ser um ótimo
7610 DATA "mês para você em casa, e"
7620 DATA você se envolverá em
7630 DATA acirrados debates políticos
7640 DATA Se você vai se mudar em 1988
```

MICRO DICAS

COMO APERFEIÇOAR O PROGRAMA

Embora seja muito divertido o programa de horóscopo apresenta uma séria desvantagem: não renova as interpretações e previsões, a não ser por um sorteio muito restrito. Assim, ele "perde a graça" depois de uma ou duas consultas.

Se você conhece um pouco de programação BASIC, porém, poderá modificar o programa e tornar seu interesse mais duradouro. Eis aqui algumas práticas sugestões:

— aproveite a idéia adotada em nosso programa original e aumente o número de alternativas de interpretação do horóscopo para cada signo. Se você colocar oito ou dez alternativas mais curtas para cada signo, e usar o gerador de números aleatórios para escolher uma delas, diminuirá a chance de que o programa se repita;

— além do signo do zodíaco (que corresponde à constelação que está no ponto mais alto do céu, no momento do nascimento), existe o signo ascendente (que é a constelação que está apontando no horizonte no momento do nascimento). Inclua-o no programa, com o conjunto de interpretações correspondente;

— armazene os textos de interpretação em um arquivo de disco, separadamente do programa. Depois que o signo da pessoa foi determinado, o texto correspondente será carregado na memória e mostrado no vídeo. Você poderá, assim mudar constantemente os textos, sem precisar alterar o programa propriamente dito.

```
7650 DATA considere cuidadosamente todas
7660 DATA "as implicações especialmente"
7670 DATA durante Abril e Agosto
7680 DATA Nas duas últimas semanas de Mai
7690 DATA você vai fazer pequenas viagens
7700 DATA para visitar velhos amigos
7710 DATA " "
7720 DATA Existe a possibilidade de uma
7730 DATA viagem em Julho com alguém que
7740 DATA não necessariamente seu parceir
7750 DATA " "
7760 DATA Um amor de verão pode acabar
7770 DATA se você tornar se muito
7780 DATA possessivo Você pode ser
7790 DATA envolvido por alguém frívol
7800 DATA "Para as mulheres, a amizade"
7810 DATA baseada na figura paterna pode
7820 DATA tornar se muito profunda
7830 DATA " "
```

AVALANCHE: COMEÇA O JOGO

- ROTINA FINAL
- ORDEM DE CHAMADA
- RECOMPENSA, MORTE OU VIDA
- CÂMARA LENTA
- INTERRUPÇÃO

Na base da encosta, Willie permanece em compasso de espera. O cenário está completo — com pedras, buracos, cobras e mar. Para que o jogo comece, falta só um comando: "Ação!"

Até agora, você digitou e testou separadamente as várias rotinas que compõem o videogame *Avalanche*. Neste artigo, fornecemos a rotina final, que chama todas as outras na ordem certa e executa o jogo completo.

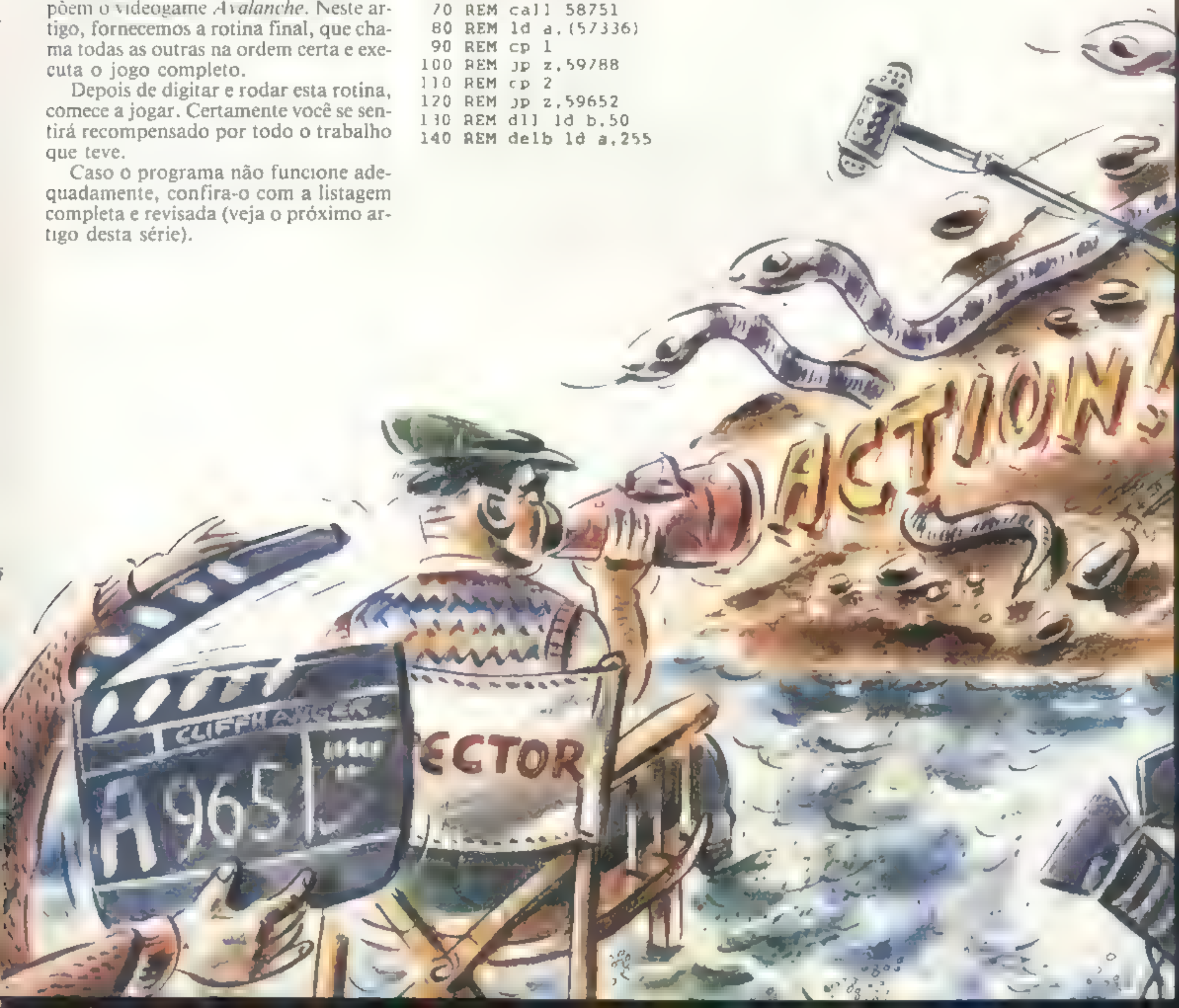
Depois de digitar e rodar esta rotina, comece a jogar. Certamente você se sentirá recompensado por todo o trabalho que teve.

Caso o programa não funcione adequadamente, confira-o com a listagem completa e revisada (veja o próximo artigo desta série).

S

A rotina a seguir é o laço principal que completa o jogo:

```
 10 REM org 58702
 20 REM alp call 59153
 30 REM call 58993
 40 REM call 59823
 50 REM call 58882
 60 REM call 58795
 70 REM call 58751
 80 REM ld a,(57336)
 90 REM cp 1
100 REM jp z,59788
110 REM cp 2
120 REM jp z,59652
130 REM dll ld b,50
140 REM delb ld a,255
150 REM dela dec a
160 REM jr nz,dela
170 REM djnz delb
180 REM ld a,254
190 REM in a,254
200 REM bit 0,a
210 REM jr nz,alp
220 REM ret
```



Quando essa rotina e o restante do programa estiverem na memória, comece o jogo digitando a instrução:

```
LET L=USR 58576
```

Como você deve ter notado, ela chama o endereço do rótulo **gbin**, que está na origem da rotina de inicialização do jogo, no artigo da página 969.

ORDEM DE CHAMADA

A rotina principal chama, nesta ordem, a rotina de movimentação de Willie, em 59153; a rotina das pedras, em 58993; a rotina das cobras, em 59823; a rotina do mar, em 58882; a rotina da nuvem, em 58795, e a rotina das gaivotas, em 58751.

RECOMPENSA, MORTE OU VIDA

Em seguida, a rotina verifica a chamada variável da morte. Ela é transferida de 57336 para o acumulador e comparada com 1. Se este for o seu valor, a instrução **jr z** manda o processador para a rotina da recompensa, no endereço 59788. Essa rotina incrementa o escore e coloca uma nova tela.

O conteúdo do acumulador é então comparado com 2. Se ele contém esse valor, o processador é mandado para a rotina da morte, no endereço 59652. Esta é a rotina que enterra Willie e encerra a partida.

CÂMARA LENTA

Se o processador fosse instruído só para executar repetidamente a rotina final, a cada vez ele chamaria todas as outras rotinas na memória, e o jogo seria tão rápido que se tornaria impraticável. Assim, para tornar o movimento mais lento, dois laços são construídos, o que atrasa o processador em dois centésimos de segundo. Pode não parecer muito, mas, considerando-se que a rotina é chamada seguidamente, trata-se de um tempo significativo.

B é carregado com 50, e A, com 255. A seguir, o conteúdo de A é decrementado. A instrução **jr nz** manda o processador para o laço **dela** 256 vezes, para que A seja decrementado até 0.

A instrução **djnz** decrementa o conteúdo do registro B e faz o processador voltar e carregar A com 255 de novo. Isso é feito até que B acabe sendo reduzido a 0. Portanto, o laço externo é executado cinquenta vezes, e o interno, 255 x 50 vezes.

O registro B contém o valor 50 apenas quando o jogo começa. À medida que Willie recupera seu lanche, um novo valor é colocado no endereço ocupado por 50 na instrução **ld b**. Se você voltar ao programa dos prêmios, verá que o valor nesse endereço é carregado em A, decrementado e colocado de volta em B cada vez que Willie recupera um item do seu lanche. Essa operação resulta, efetivamente, em uma alteração no programa — ou seja, uma rotina altera outra rotina. Tal expediente só é recomendável quando seus efeitos forem totalmente controlados.

O processador executará o laço de atraso uma vez menos, acelerando o jogo em cerca de 90 microssegundos.

FORA DO LAÇO

Como todo bom programa, este também oferece ao usuário a alternativa de interromper a execução, sem ter que desligar o computador e perder tudo o que está na memória.

Neste ponto do programa, precisa-



mos verificar se a tecla <BREAK> foi pressionada. Para isso, usamos o comando **in**. Se <BREAK> não foi pressionada, a instrução **jr nz, alp** faz o processador iniciar o laço principal outra vez. Se foi, o processador encontra a instrução **ret** e volta para o BASIC.

T 

O programa a seguir é o laço principal que completa o jogo.

```
10   ORG 20932
20   ALP LDA #5
30   STA 18258
40   CLR 18261
50   JSR ELB
60   BLP JSR MAN
70   JSR BAR
80   JSR SNK
90   JSR SEA
100   JSR MOVSUN
110   LDA 18252
120   CMPA #1
130   LBEQ RWD
140   CMPA #2
150   LBEQ DIE
160   DLL LDB #100
170   DEL CLRA
180   DELA DECA
190   BNE DELA
200   [illegible]
210   BNE DEL
220   JSR 41409
230   CMPA #3
240   BNE BLP
250   RTS
260   MOVSUN EQU $4D0F
270   ELB EQU $4B59
280   MAN EQU $4DBE
290   SNK EQU $5178
300   SEA EQU $4CDE
310   RWD EQU $50F1
320   DIE EQU $5050
330   BAR EQU $4045
```

Carregue a parte já digitada de *Avalanche* e monte esse programa no topo.

```
10   ORG 19572
20   JMP ALP
30   ALP EQU $51C4
```

Tecle a instrução.

```
EXEC 19426
```

e o jogo irá começar!

PREPARANDO A ROTINA

A é carregado com 5, valor armazenado em 18258, para ajustar o atraso do sol. A posição de memória 18261, que contém a variável do salto, é carregada com 0. Garantimos, assim, que Willie não apareça saltando na tela.

O processador vai para a rotina **ELB** e define os bits necessários para os níveis superiores do jogo. Segue, então, para a rotina **MAN**, que cuida do movimento de Willie e, em seguida, salta para **BAR**, **SNK**, **SEA** e **MOVSUN** — rotinas que movimentam, respectivamente, a pedra, as cobras, o mar e o sol.

O conteúdo da variável morte, em 18252, é carregado no acumulador e comparado com 1. Se seu valor for igual a 1, Willie conquistou um prêmio. **LBEQ** faz o processador executar um longo desvio para a rotina que oferece a Willie sua recompensa.

Se a variável morte não for igual a 1, **CMPA #2** verifica se seu valor é 2. A presença desse número indica que Willie está morto — o processador se desvia então para a rotina **DIE**, que toma todas as providências necessárias para os funerais.

ROTINA DE ATRASO

B é carregado com 100, funcionando como contador para o maior laço de atraso. Note que a posição de memória ocupada por 100 quando o programa é montado — $51EE – equivale ao byte decrementado na rotina do escore.

A é apagado e decrementado, o que torna seu conteúdo, 255 igual a -1 na representação binária. A instrução **BNE** verifica se A chegou a 0; se ainda não chegou, é novamente decrementado. Caso contrário, o processador sai fora desse laço. Em seguida, decrementa B e salta para o laço interno, onde A é decrementado. Isso é feito até que B tenha chegado a 0.

Em outras palavras: no início, o processador fica no laço 256 x 100 vezes, para tornar a execução mais lenta. Porém, quando Willie consegue uma recompensa, o jogo se acelera, pois o processador fica nesse laço apenas 256 x 99 vezes, ou 256 x 98 ou 256 x 97 e assim por diante.

FORA DO LAÇO

Para examinar o teclado, saltamos para a sub-rotina em 32774. Verificamos então se a tecla **<BREAK>** foi pressionada, comparando o valor recebido no acumulador com 3.

Se a tecla não foi pressionada, a ins-



trução **BNE** volta para dar prosseguimento ao jogo. Caso contrário, o processador continua, encontra **RTS** e retorna para o BASIC.

FECHE O CÍRCULO

Acrescentamos um último detalhe ao programa. Na rotina que seleciona a tela apropriada e ajusta o escore havia, originalmente, um **RTS** e duas instruções **NOP**. Elas foram usadas para reservar o espaço em que colocaríamos uma instrução de salto, que iria funcionar quando o jogo fosse executado pela primeira vez. Não podíamos introduzir essa instrução antes de escrever a rotina de acionamento. Agora que o programa está pronto, montamos um **JMP ALP** naquele endereço. Com isto, fechamos o círculo, completando *Avalanche*, nosso videogame em Assembler.



O programa a seguir é o laço principal que completa o jogo:

```
org 54101
jg call 54603
call -11136
call 55564
call -11318
call -11266
call -11380
ld a,(-5201)
cp 1
 jp z,55506
 cp 2
 jp z,55334
 ld b,50
 atr ld a,255
 dl dec a
 jr nz,dl
 djnz atr
 ld a,7
 call 321
 bit 4,a
 jr nz,jg
 ret
 end
```

Quando essa rotina e o restante do programa já estiverem na memória, dê início ao jogo executando o pequeno programa BASIC:

```
10 DEFUSR= -11681
20 A=USR(0)
30 END
```

Como você deve ter notado, esse programa chama o endereço inicial da rotina que inicializa o jogo, publicada no artigo da página 969.

ORDEM DE CHAMADA

O laço principal chama, nesta ordem, a rotina de movimentação de Willie, em 54603; a rotina das pedras, em -11136; a rotina das cobras, em 55564; a rotina do mar, em -11318; a rotina da nuvem, em -11266, e a rotina das gaivotas, em -11380.

PRÊMIO, MORTE OU VIDA

Em seguida, a rotina verifica a chamada variável da morte. Ela é transferida de -5201 para o acumulador e comparada com 1. Se este for o seu valor, a instrução **jp z** manda o processador para a rotina da recompensa, no endereço 55506. Essa rotina aumenta os pontos de Willie e seleciona um nível maior de dificuldade.

O conteúdo do acumulador é comparado com 2. Se ele contém esse valor, o processador é mandado para a rotina da morte, em 55334. Essa rotina enterra Willie e encerra a partida.

CÂMARA LENTA

Se instruíssemos o processador para executar continuamente esse laço, a cada volta ele chamaria todas as rotinas da memória com crescente velocidade, tornando o jogo impraticável. Para tornar a execução mais lenta, montamos dois laços de atraso, como já fizemos na rotina da morte.

B é carregado com 50, e A, com 255. A seguir, o conteúdo de A é decrementado. A instrução **jr nz** faz o processador continuar a decrementar A, no laço **dl**, até que seu valor seja 0.

A instrução **djnz** decrementa o conteúdo do registro B e faz o processador voltar ao rótulo **atr** e carregar A com 255 de novo. Isso é feito até que B se reduza a 0. Portanto, o laço externo é executado cinqüenta vezes, e o laço interno, 255 x 50 vezes.

O registro B contém o valor 50 só quando o jogo começa. À medida que Willie recupera seu lanche, um novo valor é colocado no endereço ocupado por 50 na instrução **ld b.** Se você voltar à rotina do escore, verá que o valor nesse endereço (54133) é decrementado cada vez que Willie recupera uma parte de seu lanche. Essa operação resulta, efetivamente, em uma alteração no próprio programa — ou seja, uma rotina altera outra rotina. Esse expediente só é recomendável quando seus efeitos forem controlados.

Diminuindo o valor carregado em B, o processador executará o laço interno 256 vezes menos, acelerando o jogo.

FORA DO LAÇO

Como todo bom programa, este também oferece ao usuário a alternativa de interromper a execução, sem ter que desligar o computador e perder tudo o que está na memória.

Neste ponto do programa, precisamos ver se <STOP> foi pressionada. Essa operação é idêntica à que executamos para checar se M ou N foram pressionadas na rotina de movimentação de Willie.

O número 7 é carregado no acumulador e a rotina 321 da ROM é chamada. O número em A corresponde à linha da matriz do teclado na qual está a tecla que desejamos verificar. Se alguma tecla dessa linha foi pressionada, a rotina 321 coloca o valor 1 no bit do acumulador correspondente a ela.

A instrução **bit 4,a** examina o bit 4 do acumulador, que corresponde à tecla <STOP>. Se ela não foi pressionada, a instrução **jr nz, jg** faz o processador reiniciar o laço principal, **jg.** Caso contrario, o processador encontra a instrução **ret** e volta para o BASIC.



O PINTOR ALOPRADO

- OBJETIVO DO JOGO
- O ROLO E A TINTA
- SOMA DE VERIFICAÇÃO
- NÍVEIS DE DIFICULDADE
- ESCORE



Este divertido jogo é um bom treino para os pintores "de fim de semana". Manejando o rolo, eles devem impedir que a tinta escorra pela parede e manche seu precioso carpete...

Manchar o carpete de tinta é um dos pesadelos dos pintores domésticos. Eis aqui uma boa oportunidade para que aprimorem sua técnica de trabalho e deixem de fazer tanta sujeira e provocar tanta confusão na casa.

O objetivo do jogo é impedir que a tinta, que escorre pela parede (a tela), chegue até o chão (linha inferior da tela). Para isso, você deve movimentar um rolo de pintura (um sinal gráfico horizontal, na tela), usando as teclas de controle do cursor.

VERSÕES DO PROGRAMA

O programa é apresentado em duas versões: uma para os microcomputadores compatíveis com a linha Sinclair Spectrum (TK 90X), e outra, para os da linha TRS-Color. Ambas utilizam código de máquina, para que os cálculos sejam efetuados com a rapidez necessária.

Os programas em código de máquina estão incluídos no programa em BASIC, na forma de listagens numéricas decimais, dentro de linhas **DATA**. O programa de carregamento vê se os códigos foram digitados corretamente, por meio de somas de verificação (*checksum*). Apesar disso, convém gravar o programa em fita ou disco antes de usá-lo, pois um só código errado poderá danificar o conteúdo da memória.

```
1 CLEAR 28761: GOSUB 100
5 CLS : PRINT AT 8,,," ESCOLHE
 NIVEL DE DIFICULDADE"'TAB 8
,"1- MUITO FACIL"'TAB 8,"2
- FACIL"'TAB 8,"3- NORMAL"'
TAB 8,"4- DIFICIL"'TAB 8,"
5- IMPOSSIVEL"
6 LET D$=INKEY$: IF D$<"1"
 OR D$>"5" THEN GOTO 6
 7 POKE 28951,((D$="1")*200)+
 ((D$="2")*175)+((D$="3")*80)
 +((D$="4")*40)+(D$="5")
10 BORDER 0: PAPER 7: INK 2:
CLS : LET a$="": FOR n=1 TO
32: LET a$=a$+" ": NEXT n
14 POKE 28953,0: FOR n=1 TO 4
: PRINT PAPER 2;a$;: NEXT n
15 FOR N=19 TO 21: PRINT
PAPER 6;AT N,0;A$: NEXT N
16 PLOT 0,143: DRAW 255,0
20 RAND USR 28672: RAND USR
28702
40 PRINT AT 12,7; FLASH 1;
PAPER 5; INK 0;" F I M   D E
 J O G O "'' FLASH 0;AT 14,7
; PAPER 7;"PLACAR FINAL ";
LET B$="": FOR N=28945 TO
28950: LET B$=B$+CHR$ (PEEK N
): NEXT N: PRINT B$
45 FOR N=1 TO 500: NEXT N
50 IF INKEY$="" THEN GOTO 50
60 RUN 5
100 LET L=500: RESTORE L: FOR
N=28672 TO 28961 STEP 8
110 LET T=0: FOR D=0 TO 7:
READ A: POKE N+D,A: LET T=T+A:
NEXT D
120 READ A: IF A<>T THEN
PRINT "ERRO NOS DADOS DA LINHA
 ";L: STOP
130 LET L=L+10: NEXT N: RETURN
500 DATA 33,34,113,6,0,62,32,
119,399
510 DATA 35,16,252,33,128,100,
34,32,630
520 DATA 113,33,48,48,34,17,
113,34,440
530 DATA 19,113,34,21,113,201,
205,228,934
540 DATA 112,89,22,0,33,34,113
,25,428
550 DATA 126,60,254,156,200,
245,229,205,1475
560 DATA 176,34,209,193,245,
126,254,255,1492
570 DATA 40,2,120,18,241,254,0
,71,746
580 DATA 62,128,40,4,203,31,16
,252,736
590 DATA 70,176,119,58,25,113,
214,64,839
600 DATA 50,25,113,194,194,112
,237,75,1000
610 DATA 42,113,62,0,33,26,113
,197,576
620 DATA 205,214,112,193,62,
223,219,254,1482
630 DATA 245,203,31,203,31,48,
16,203,980
640 DATA 31,48,15,203,31,48,14
,203,593
650 DATA 31,48,13,241,195,156,
112,12,808
660 DATA 24,7,4,24,4,5,24,1,93
670 DATA 13,241,121,254,240,48
,13,120,1050
680 DATA 254,150,48,8,254,32,
56,4,806
690 DATA 237,67,32,113,237,75,
32,113,906
700 DATA 197,33,29,113,205,214
,112,193,1096
710 DATA 121,230,248,79,89,22,
0,33,822
720 DATA 34,113,25,72,6,24,126
,185,585
730 DATA 32,5,61,119,205,252,
112,35,821
740 DATA 16,244,42,23,113,45,
32,253,768
750 DATA 37,242,197,112,62,127
,219,254,1250
760 DATA 203,31,218,30,112,201
,229,120,1144
770 DATA 205,176,34,235,225,1,
3,0,879
780 DATA 237,176,235,201,42,
118,92,237,1338
790 DATA 91,120,92,25,237,90,
84,93,832
800 DATA 41,41,25,41,41,41,25,
34,289
810 DATA 118,92,76,201,213,245
,17,22,984
820 DATA 113,26,60,254,58,32,6
,62,611
830 DATA 48,18,27,24,244,18,
241,209,829
840 DATA 201,48,48,48,48,48,48
```

```
,200,689
850 DATA 0,64.0.0.0.255.255.
,55 922
860 DATA .28.10..36.36 3'.37.
.5. 5,444
```

O jogador controla o rolo por meio das teclas Y, U, I e O. O programa em código de máquina é responsável pelo movimento do rolo e dos pingos de tinta, além de cuidar do escore (número de pontos marcados).

No programa em BASIC, a linha 1 reserva espaço para o programa em código de máquina, por intermédio do comando **CLEAR**, e chama a rotina que começa na linha 100. Esta lê os códigos numéricos correspondentes ao programa em linguagem de máquina e os coloca na memória reservada (usando comandos **POKE**, na linha 110).

SOMA DE VERIFICAÇÃO

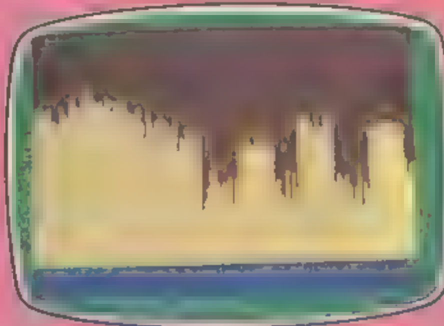
A soma de verificação é armazenada na variável **T**, que é checada a cada oito códigos pela linha 120 do programa. Essa linha informa se o valor da soma é igual ao primeiro número lido na linha **DATA**. Em caso afirmativo, o programa continua até ler todos os códigos, que estão armazenados da linha 500 à linha 860.

As linhas 5 a 7 inicializam o nível de dificuldade — que pode ser fácil, razoavelmente fácil, normal ou difícil — e colocam o valor correspondente de atraso de tempo nas locações absolutas do programa em código de máquina, por meio de comandos **POKE**.

As linhas 10 a 16 montam a tela gráfica e a 20 chama a rotina em linguagem de máquina. As linhas 40 a 60 finalizam o jogo e imprimem o escore.

T

```
10 CLEAR 200.15799 CLS
20 FOR K=0 TO 13 T=0 FOR J=0 TO
 25:READ A:T=T+A
30 POKE 15800+K*26+J,A
40 NEXT READ A IF T<>A THEN PRI
NT"ERRO NOS DADOS DA LINHA",100
0+K*10:END ELSE NEXT
50 CLS PRINT @2,"ESCOLHA NIVEL
DE DIFICULDADE"
60 PRINT @200,"1 - FACIL":PRINT
 @232,"2 - SIMPLES":PRINT @264,
"3 - MEDIO":PRINT @296,"4 - DIF
ICIL":PRINT @328,"5 - IMPOSSIVE
L"
70 AS=INKEYS IF AS<"1" OR AS>"5
" THEN 70
80 LV=VAL(AS) POKE 16162,6 LV P
OKE 16.64.128+64*(LV>2) POKE 16
```



A tinta escorrendo na tela do TRS-Color.

```
167.RND(256)-1
90 PMODE 3.1:PCLS 2:SCREEN 1.0
100 COLOR 4 LINE(0,0) (2255.0).
PSET COLOR 3 LINE(0,168)-(255,1
91),PSET.BF
110 DEFUSR0=15800 S=USR0(0)
120 SC=0 CLS FOR K=5 TO 0 STEP
-1 SC=SC*256+PEEK(16173+K) NEXT
130 PRINT @8,"VOCE GANHOU".SC
140 PRINT @161."QUALQUER TECLA
PARA RECOMECAR":AS=INKEYS
150 IF INKEYS="" THEN 150 ELSE
50
1000 DATA 127,63,33.127,63,37.7
9.95.253,63,45,253,63,47,253.63
.49.142,19,14,191,63,51,158.186
.48,2585
1010 DATA 137,1.0,191,63.41,48.
137.19.223,191,63.43,204.0.128.
142.63.53,167,128.90.38.251,141
.29.2591
1020 DATA 182.63,37.176.63.36.1
83,63,37.38.243.141.119,23.0.14
5,190.63.34.48.31.38.252.125.63
.33.2426
1030 DATA 39,226,57.206,63,53,1
41,59.196.127,52.4.51,197.166,1
96,198.32,61.211,186.31,1.53,4.
31.2641
1040 DATA 152,84,84,58,230,132.
38.1,57,132,3,64,139,3,198.3,74
.43.4.88,88,32,249.234,132.231.
2553
1050 DATA 132,108.196.166.196.1
29.168,37,5,134,1,183,63 33,57.
190.63,38./9,95,179.63.38.36,2.
48.2439
1060 DATA 31,179.63.38,36,2.48.
31,195.255.254.36,2.48.1,52.16.
163.225,37.3.131.0,1.253.63.216
3
1070 DATA 38.57,134.247.127,63.
40,120.63,40.183.255,2,246.255.
0,193.247.38,3,124.63.40,26.1.7
3,2678
1080 DATA 129.127,34,213.57,190
.63,51.198.3.134.85.167,128.90.
38.251.190.63.51.116.63,40.36.1
0.31,2579
1090 DATA 16.203.3,196.3..39.2.
48.1.116.63.40.36.8.31.16.196.3
1.39.2.48.31.116.63.40.36.1451
1100 DATA 8,188.63,43.34,3.48.1
36.32.116,63.40.36.14,168.63.41
.37,9.48.136,224.52.16,141,13.1
792
1110 DATA 53,16.191,63,51.198.3
.111.128,90,38.251.57,31,16.142
.63.53,147,186.52.4.196,31.88.8
8.2347
1120 DATA 48,133.53.4.88,73,88.
73.31.88.73.137.92,134.12.225.1
32.38.4.106,132.141.6.48.1.74.2
034
1130 DATA 38.243.57.52.2.,142.6
3.45.198.6.26.1.166.132.137.0.1
67,128.90.38.247,51.150.1.3.0.2
205
```

O jogador controla o rolo de tinta pelas teclas de movimentação do cursor (flechas). O programa em código de máquina é responsável pelo movimento do rolo e dos pingos de tinta, além de cuidar do escore (número de pontos marcados). Ele verifica se alguma das teclas de controle do cursor está sendo pressionada e desloca o rolo de tinta na direção indicada. Se o jogador consegue deslocar um pingo de tinta, um atraso de tempo é introduzido e o escore é aumentado.

No programa em BASIC, a linha 10 reserva espaço para o programa em código de máquina através do comando **CLEAR**. As linhas 20 a 40 lêem os códigos numéricos correspondentes ao programa em linguagem de máquina e os coloca na memória reservada (usando comandos **POKE**, na linha 30).

A soma de verificação é armazenada na variável **T**, que é checada pela linha 40 do programa. Essa linha informa se o valor da soma é igual ao primeiro número lido na linha **DATA**. Em caso afirmativo, o programa continua até ler todos os códigos, que estão armazenados nas linhas 1000 a 1130.

NÍVEL DE DIFICULDADE

As linhas 50 a 80 inicializam o nível de dificuldade — que pode ser simples, médio, difícil e impossível — e colocam o valor correspondente de atraso de tempo nas locações absolutas do programa em código de máquina, por meio de comandos **POKE**.

A linha 100 monta a tela gráfica e a linha 110 chama a rotina em linguagem de máquina.

As linhas 120 a 150 finalizam o jogo e imprimem o escore. A seguir, o programa pergunta ao jogador se ele deseja participar de mais uma rodada.

OPERAÇÕES COM DATAS

- CONVERSÃO DO FORMATO
- COMPRESSÃO DE DATAS
- TESTE DE VALIDADE
- DATA CORRIDA
- INTERVALO ENTRE DATAS

A armazenagem e a manipulação de datas em BASIC apresentam problemas nem sempre fáceis de resolver. Com as rotinas aqui fornecidas, você simplificará bastante seu trabalho.

Mais cedo ou mais tarde, o programador vai ter que trabalhar com cálculos de datas. Como no BASIC não há nenhuma função que facilite a realização desses cálculos, a tarefa poderá envolver algumas complicações, devido à própria irregularidade do sistema de datação em uso — meses com número diferente de dias, anos bissextos etc.

Os principais problemas de cálculo e manipulação de datas, que surgem sobretudo em programas para aplicações financeiras e/ou comerciais, são:
— representação interna de datas na memória do computador;
— checagem da validade de uma data;
— determinação do dia da semana para uma data e do número de dias compreendido entre duas datas;
— cálculo de data corrida a partir de certa data de calendário.

Mostraremos em *INPUT* alguns truques que simplificam essas manipulações.

ARMAZENAGEM DE UMA DATA

Há diversos tipos de notação para datas. A mais comum é a *data gregoriana*, ou data de calendário, que tem a forma: dia/mês/ano (D/M/A).

Essa data pode ser armazenada no micro de diferentes maneiras. A mais direta usa o *formato de oito bytes*:

DD/MM/AA

Em conseqüência, deve ser armazenada em uma variável literal (alfanumérica), e não em uma variável numérica, ocupando, ao todo, oito bytes.

Podemos reduzir o espaço para a armazenagem suprimindo os sinais de separação (barras ou pontos), já que é possível inseri-los novamente no momento de exibir ou imprimir uma data. Temos, então, o *formato de seis bytes*:

DDMMAA

Armazenando as datas como são visualizadas, é impossível ordená-las. Colocar um conjunto de datas em ordem ascendente, por exemplo, resultará em uma enorme confusão: uma data como 010187 virá antes de 011286.

Por essa razão, é preferível usar o formato sueco, já adotado universalmente em sistemas de computação:

AAMMDD

No exemplo acima, 861201 aparecerá, corretamente, antes de 870101.

Eis aqui duas rotinas para converter uma notação em outra. A variável **N$** representa uma data em formato normal, de oito bytes; **I$**, uma data em formato invertido, de seis bytes.

Conversão de formato normal para formato invertido:

```
1000 I$=MID$(N$,7,2)+MID$(N$,4,
2)+MID$(N$,1,2) RETURN
```

```
1000 LET I$=N$(7 TO)+N$(4 TO 5
)+N$(1 TO 2)
1010 RETURN
```

Lembre-se de dimensionar **N$(8)** e **I$(6)** no começo do programa.

Conversão de formato invertido para formato normal:

```
1100 N$=MID$(I$,5)+"/"+MID$(I$,
3,2)+"/"+MID$(I$,1,2):RETURN
```

```
1100 LET N$=I$(5 TO)+"/"+I$(3
TO 4)+"/"+I$(1 TO 2)
1110 RETURN
```

Seis bytes podem significar muito espaço de memória para certas aplicações, por exemplo, para um banco de dados, em que cada registro deve reservar espaço para uma ou mais datas.

Podemos reduzir ainda mais o espaço de memória ocupado por uma data de calendário, mas isso envolve algumas manipulações que codificam a data, impedindo sua exibição imediata (sem a prévia decodificação).

Um dos formatos codificados de datas é o *formato de três bytes*:

```
CHR$(AA)+CHR$(MM)+CHR$(DD)
```

A data será armazenada em uma cadeia literal de três bytes, no formato sueco; mas não poderá ser impressa ainda. Seguem-se duas rotinas que realizam as conversões. Nelas, **D$** é uma variável literal com uma data em oito bytes, e **C$**, uma variável com uma data em formato comprimido de três bytes.

Compressão de oito para três bytes

```
1200 C$=CHR$(VAL(MID$(D$,7,2)))
+CHR$(VAL(MID$(D$,4,2)))+CHR$(V
AL(MID$(D$,1,2))):RETURN
```

```
1200 LET C$=CHR$ VAL D$(7 TO 8)
+CHR$ VAL D$(4 TO 5) + CHR$ VAL
 D$(1 TO 2)
1210 RETURN
```

Não se esqueça de dimensionar **C$(3)** e **D$(8)**, no começo do programa.

Descompressão de três para oito bytes.

```
1300 D$=RIGHT$(STR$(ASC(MID$(C$
,2))),2)+"/"+RIGHT$(STR$(ASC(MI
D$(C$,3))),2)+"/"+RIGHT$(STR$(A
SC(C$)),2):RETURN
```

```
1300 LET D$=STR$ CODE C$(3 TO 3
)+STR$ CODE C$(2 TO 2)+STR$ COD
E C$(1 TO 1)
1310 RETURN
```

Essa técnica tem só uma desvantagem: na armazenagem de datas comprimidas em três bytes em arquivos seqüenciais (fita ou disco), a transmissão será truncada sempre que surgir um dia 13 (pois **CHR$(13)** é o código ASCII para fim de linha). Isto não ocorrerá com arquivos de acesso aleatório.

Nada impede que se armazene uma data de calendário em formato numérico — o que é até desejável em certas aplicações. Porém será desperdício de memória, mesmo que usemos variáveis inteiras (que ocupam só dois bytes de espaço cada no TRS-80, TRS-Color, Apple, TK-2000 e MSX; o ZX-81 e o Spectrum não oferecem essa possibilidade).

A armazenagem de uma data de calendário em três variáveis inteiras — por exemplo, **D**, **M** e **A** — ocupará oito bytes. Podemos armazená-las em uma variável de precisão simples:

```
DT = A * 10000 + M * 100 + D
```

Veja os exemplos seguintes:

Data	DT
01 01/47	470.101
31/12/87	871.231

Note que **DT** não pode ser uma variável inteira, pois esta não aceitaria os números maiores. Assim, ficamos, de novo, com seis bytes por data.

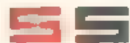
VALIDADE DE UMA DATA

Uma boa prática de programação consiste em testar a validade da data de calendário que foi entrada pelo teclado, checando separadamente o dia, o mês e o ano.

Eis aqui uma rotina simples para teste de uma data entrada no formato de oito bytes (variável **D$**):



```
1500 E=0:M$="31293130313031313O
313031"
1510 IF LEN(D$)<>8 THEN E=1:RET
URN
1520 M=VAL(MID$(D$,4,2)):IF M<1
OR M>13 THEN E=1·RETURN
1530 D=VAL(MID$(D$,1,2)):IF D<1
OR D>VAL(MID$(M$,(M-1)*2+1,2))
THEN 1590
1550 RETURN
```



```
1500 LET M$="31293130313031313O
313031"
1505 LET E=0
1510 IF LEN(D$)<>8 THEN GOTO 15
60
1520 LET M=VAL D$(4 TO 5)
1525 IF M<1 OR M>13 THEN GOTO 1
560
1530 LET D=VAL D$(1 TO 2)
1535 IF D<1 OR D>VAL M$((M-1)*2
+1 TO (M-1)*2+2) THEN GOTO 1590
1550 RETURN
1590 LET E=1
1600 RETURN
```

Se a variável **E** retornar igual a 0, a data é válida; se retornar igual a 1, houve erro de entrada.

A DATA CORRIDA

Para muitas das técnicas de cálculo de funções relacionadas a datas, é preciso saber o dia do ano de certa data de calendário. Isso caracteriza a data corrida, composta do ano e do dia do ano (um número de 1 a 366). Por exemplo, a data corrida para 14 de maio de 1981 é 134/81.

Esta sub-rotina calcula a data corrida a partir de uma data de calendário **D$**, em formato de oito bytes. O resultado será armazenado em **N**.



```
1600 M$="0000310590891201501812
12242273303334"
1605 D%=VAL(MID$(D$,1,2))·M%=VA
L(MID$(D$,4,2)):A%=VAL(MID$(D$,
7,2))
1610 N%=VAL(MID$(M$,(M%-1)*3+1,
3))+D%
1620 IF M%>2 AND (A% AND NOT-4)
=0 THEN N%=N%+1
1630 RETURN
```



```
1600 LET M$="000031059089120150
181212242273303334"
1605 LET D=VAL D$(1 TO 2)
1606 LET M=VAL D$(4 TO 5)
1607 LET A=VAL D$(7 TO 9)
1610 LET N=VAL M$((M-1)*3+1 TO
(M-1)*3+3)+D
1620 IF M>2 AND (A AND NOT-4)=0
 THEN LET N=N+1
1630 RETURN
```

A variável **M$**, na linha 1600, contém o número do dia no ano equivalente ao primeiro dia de cada mês, menos 1, para ano não bissexto. As variáveis **D**, **M** e **A** são extraídas da variável **D$**, que contém a data de calendário. Na linha 1610, o número de dias correspondente a data **D,M,A** é calculado somando-se **D** ao número que se obtém extraindo-se do *string* **M$** o valor para o primeiro dia do mês **M**.

Finalmente, a linha 1620 verifica se o mês é março ou um dos meses que o sucedem e se o ano é divisível por 4 (ano bissexto). Nesse caso, acrescenta-se 1 ao número do dia, para compensar o fato de fevereiro ter 29 dias. O emprego, pouco usual, da expressão lógica **AND NOT** funciona como teste do resto da divisão por 4.

A rotina só pode ser usada para os anos de 1901 a 1999, pois o cálculo de anos bissextos não funciona para seculos ímpares — a não ser que o ano em questão seja divisível por 400.

DIAS ENTRE DUAS DATAS

O modo mais fácil de determinar o número de dias transcorridos entre duas datas de calendário consiste em calcular a data corrida de cada uma e obter a sua diferença mais 1. A operação funcionará bem se as datas forem do mesmo ano; caso contrário, dará resultados errados.

Por isso, devemos obter um outro número (chamado *data juliana*), que leva em conta o total de dias entre uma data-base, fixa, e a data de calendário que especificamos. Para simplificar o cálculo, multiplicamos o ano por 365, de forma que a data-base passa a ser 1900; depois, somamos o número de dias referente aos anos bissextos entre 1900 e o ano atual.

A próxima sub-rotina calcula a diferença entre duas datas, entre 1901 e 1999. Adicione-a à sub-rotina anterior, que começa na linha 1600.



```
1700 D$=D1$ GOSUB 1600 N1=N%+
A%*365+INT((A%-1)/4)
1710 D$=D2$ GOSUB
1600·N2=N%+A%*365+INT((A%-1)/4
1720 P=INT(N2 N1) RETURN
```



```
1700 LET D$=D1$
1705 GOSUB 1600
1706 LET N1=N+A*365+INT (A-1)/4
1710 LET D$=D2$
1715 GOSUB 1600
1716 LET N2=N+A*365+INT (A-1)/4
1720 LET P=INT N2-N1
1725 RETURN
```

A sub-rotina aceita como argumentos duas datas, **D1$** e **D2$**, no formato de oito bytes (DD/MM/AA), e retorna o resultado armazenado em **P**.

Para testá-la, acrescente as seguintes linhas:



```
10 PRINT "PRIMEIRA DATA (DD/MM/
AA) ";
20 INPUT D1$
30 PRINT "SEGUNDA DATA (DD/MM/
AA) ";
40 INPUT D2$
50 GOSUB 1700
60 PRINT P
70 GOSUB 10
```

Use o programa para saber quantos dias você viveu até hoje!

FERRAMENTAS PARA O SPECTRUM

- COMANDOS EXTRAS
- RENUMERAÇÃO DE LINHAS
- AUTONUMERAÇÃO DE LINHAS
- ELIMINAÇÃO EM BLOCO
- OUTROS COMANDOS

Este conjunto de ferramentas de programação BASIC tornará seu trabalho mais fácil. Com ele, você terá acesso a diversos comandos inteiramente novos no Spectrum.

Embora todos os micros examinados em *INPUT* utilizem a mesma linguagem BASIC, você já deve ter percebido que há diversas variantes, ou "dialetos" da mesma. De fato, é muito raro encontrar um programa, mesmo curto, que possa ser executado sem modificações em qualquer linha de computadores. Com freqüência, trata-se apenas de uma variação na sintaxe dos comandos ou na maneira como eles são usados; nesse caso, a adaptação do programa não envolve maiores complicações. Algumas vezes, porém, constata-se que muitos dos comandos destinados a um computador simplesmente não existem para outros. Ainda que eles não sejam essenciais para a programação de muitas tarefas, não há dúvida de que, se fossem disponíveis, o trabalho do programador se tornaria bem mais fácil. Incluem-se nessa categoria os comandos para renumeração das linhas de um programa (**RENUM**), para eliminação de blocos de linhas (**DEL**) etc. Presentes em computadores de linhas mais recentes, como o MSX, esses comandos fazem muita falta nos micros da linha Sinclair.

O programa apresentado neste artigo adiciona vários comandos desse tipo ao interpretador dos microcomputadores da linha Sinclair Spectrum, facilitando o desenvolvimento de programas em BASIC. Os micros das linhas TRS-80, TRS-Color e MSX não precisam de um programa como este, pois já dispõem da maioria dos comandos necessários em seu interpretador BASIC.

O programa foi desenvolvido em código de máquina, de modo a coexistir, na memória, com o programa em BASIC e o interpretador.

São adicionados ao interpretador oito novos comandos e funções:

- renumeração de linhas;
- eliminação de um bloco de linhas;
- número de bytes livres na memória;
- comprimento de um programa;
- numeração automática de linhas;
- catalogador de fitas;
- conversão de hexa para decimal e vice-versa.

Todas essas funções são chamadas por intermédio do comando **RANDOMIZE USR**, seguido de um número, conforme mostramos mais adiante.

Uma vez que um novo comando tenha sido ativado, ele pedirá diversos parâmetros — tais como os números das linhas a serem apagadas — através de mensagens exibidas na tela.

O programa utiliza diversas rotinas em linguagem de máquina fornecidas em artigo posterior (*Referência Cruzada*). Assim, será necessário juntar os dois programas. As instruções para a execução dessa tarefa estão contidas dentro do programa aqui apresentado. Bastará, portanto, digitar o programa, acioná-lo com um comando **RUN** e, então, seguir a orientação dada pelas mensagens exibidas na tela. O programa informará, também, se você cometeu algum erro na digitação das linhas **DATA**, que contêm o programa em código de máquina, pois efetua automaticamente uma soma de verificação. Corrija todos os erros, antes de armazenar o programa em fita.

Os códigos conjuntos do referenciador e do extensor de comandos serão armazenados em fita sob a denominação **"TOOLKIT" CODE**.

Para carregar o programa, digite:

```
CLEAR 63488
LOAD "TOOLKIT" CODE
```



A rotina de renumeração de linhas é ativada pelo comando:

```
RANDOMIZE USR 63489
```

O programa pede que se informe o incremento de linhas a ser utilizado (um número entre 1 e 255). Para achar o comprimento de um programa BASIC residente na memória, use:

```
RANDOMIZE USR 63889
```

Para achar o número de bytes disponíveis na memória, digite:

```
RANDOMIZE USR 63860
```

A numeração automática de linhas (comando **AUTO**, em outros computadores) é ativada por:

```
RANDOMIZE USR 64154
```

Em seguida o programa pedirá o número da linha inicial (entre 1 e 9900) e o incremento (1 a 9900). Para cancelar o comando, entre dois zeros quando o número de linha aparecer na tela. A rotina terminará com uma mensagem de erro, que deverá ser ignorada.

Para apagar um conjunto de linhas de um programa, digite:

```
RANDOMIZE USR 64000
```

O programa solicita que o usuário digite os números de linha do início e do fim do bloco a ser apagado. Para acionar o catalogador de fitas, chame:

```
RANDOMIZE USR 63919
```

A seguir, a borda da tela começará a piscar. Posicione a fita a catalogar no início de um programa e pressione a tecla **PLAY** do gravador. O programa exibirá na tela informações contidas no cabeçalho do arquivo (*header*).

Finalmente, para converter números decimais para hexadecimais, use:

```
RANDOMIZE USR 64394
```

Para efetuar a conversão no sentido oposto, digite:

```
RANDOMIZE USR 64453
```

Não e preciso pressionar **<ENTER>** após entrar o número hexadecimal.

```
 5 CLEAR 63488: BORDER 0:
 PAPER 0: INK 6: CLS
10 PRINT AT 0,5: INVERSE 1;"C
ONJUNTO DE FERRAMENTAS"
12 PRINT AT 8,2;" Qualquer te
cla para carregar   programa
em codigo de maquina.": PAUSE
0
14 LOAD "CREF"CODE
```



```
15 CLS : PRINT "Pokeando codigo de maquina.     Prepare o cassete para gravar."
20 LET L=100  RESTORE L: FOR N=63489 TO 64560 STEP 16
30 LET T=0: FOR D=0 TO 15
40 READ A: POKE (N+D),A: LET T=T+A: NEXT D
50 READ A: IF A<>T THEN PRINT "ERRO DE CHECKSUM EM ";L: STOP
60 LET L=L+10
70 NEXT N
100 DATA 62,12,205,48,252,205,60,250,237,67,155,248,42,83,92,1,2019
110 DATA 0,0,126,254,128,40,9,197,205,184,25,193,3,235,24,242,1865
120 DATA 205,43,45,58,155,248,205,40,45,239,4,56,205,162,45,33,1788
130 DATA 15,39,167,237,66,48,2,207,5,33,145,248,126,60,40,32,1470
140 DATA 35,229,237,91,83,92,42,75,92,167,237,82,68,77,235,237,2079
150 DATA 177,197,229,245,204,157,248,241,225,193,234,80,248,225,24,220,3147
160 DATA 42,83,92,58,155,248,54,0,35,119,205,40,45,239,49,192,1656
170 DATA 56,42,83,92,205,184,25,42,75,92,43,167,237,82,216,235,1876
180 DATA 229,239,224,15,49,56,205,162,45,225,112,35,113,43,24,228,2004
190 DATA 201,224,228,235,236,239,246,255,0,0,10,0,229,6,4,35,2148
200 DATA 126,254,14,40,4,16,248,225,201,197,35,35,35,78,35,70,1613
210 DATA 42,83,92,217,1,1,0,217,205,149,22,43,235,167,237,66,1777
220 DATA 235,48,11,217,3,217,197,205,184,25,193,235,24,234,217,197,2442
230 DATA 217,209,42,155,248,205,169,48,235,42,104,92,35,35,115,35,1986
240 DATA 114,235,62,0,167,1,9,0,237,66,56,17,60,1,90,0,1115
250 DATA 237,66,56,9,60,1,132,3,237,66,56,1,60,209,225,229,1647
260 DATA 245,130,214,4,245,6,0,56,9,79,40,12,205,85,22,35,1387
270 DATA 24,6,237,68,79,205,232,25,193,241,197,79,6,0,9,65,1666
280 DATA 229,197,35,35,235,42,104,92,1,5,0,237,176,193,225,229,2035
290 DATA 197,239,224,164,5,58,193,164,4,224,1,3,225,192,2,56,1951
300 DATA 205,213,45,193,225,198,48,119,43,16,228,229,42,104,92,35,2035
310 DATA 35,126,225,198,48,119,241,193,245,42,83,92,167,229,237,66,2346
320 DATA 225,48,8,197,205,184,25,193,235,24,242,235,35,35,293,126,2210
330 DATA 128,119,201,62,0,205,48,252,205,26,31,33,0,0,62,0,1372
340 DATA 237,66,229,193,205,43,45,62,254,205,1,22,205,227,45,201,2240
350 DATA 42,75,92,237,75,83,92,62,0,237,66,229,62,1,205,48,1606
360 DATA 252,193,205,43,45,205,227,45,62,2,205,48,252,201,221,33,2239
370 DATA 32,255,17,17,0,175,55,205,86,5,62,3,205,48,252,221,1638
380 DATA 33,32,255,221,126,0,198,6,221,229,205,48,252,62,13,215,2116
390 DATA 62,4,205,48,252,221,225,221,35,221,126,0,254,255,40,10,2179
400 DATA 6,10,221,126,0,221,35,215,16,248,62,13,215,62,5,205,1660
410 DATA 48,252,221,33,32,255,221,78,11,221,70,12,195,163,249,205,2266
420 DATA 142,250,62,10,205,48,252,205,60,250,205,110,25,229,62,13,2128
430 DATA 215,62,11,205,48,252,205,60,250,205,110,25,193,32,16,229,2118
440 DATA 35,35,126,35,95,126,87,225,237,90,17,4,0,237,90,229,1668
450 DATA 197,62,0,237,66,218,87,252,195,89,252,6,0,197,205,95,2158
460 DATA 252,205,115,252,193,254,13,40,26,254,58,48,240,214,48,56,2268
470 DATA 236,245,4,120,254,6,32,4,5,241,24,225,241,245,198,48,2128
480 DATA 215,24,218,221,33,49,255,120,254,0,40,207,33,0,0,221,1890
490 DATA 94,0,221,86,1,241,254,0,40,7,237,90,56,13,61,32,1433
500 DATA 249,221,35,221,35,5,32,231,229,193,201,207,5,42,75,92,2073
510 DATA 237,75,83,92,237,66,192,207,9,62,10,205,48,252,205,60,2040
520 DATA 250,34,30,255,62,13,215,62,12,205,48,252,205,60,250,34,1987
530 DATA 28,255,33,48,48,34,59,255,34,61,255,237,75,30,255,205,1912
540 DATA 115,251,62,2,50,107,92,50,107,92,205,149,23,205,176,22,1708
550 DATA 62,0,205,1,22,33,59,255,6,4,126,229,197,205,129,15,1548
560 DATA 193,225,35,16,245,205,44,15,205,23,27,221,33,58,92,221,1858
570 DATA 203,0,126,32,13,42,89,92,205,167,17,62,255,50,58,92,1503
580 DATA 24,206,42,89,92,34,93,92,205,251,25,120,177,32,10,223,1715
590 DATA 254,13,40,174,205,176,22,207,1,237,67,73,92,42,93,92,1788
600 DATA 235,33,85,21,229,42,97,92,55,237,82,229,96,105,205,110,1953
610 DATA 25,32,6,205,184,25,205,232,25,193,121,61,176,40,47,197,1774
620 DATA 3,3,3,3,43,237,91,83,92,213,205,85,22,225,34,83,1425
630 DATA 92,193,197,19,42,97,92,43,43,237,184,42,73,92,235,193,1874
640 DATA 112,43,113,43,115,43,114,237,75,28,255,205,115,251,241,195,2185
650 DATA 195,250,33,62,255,126,60,254,58,40,8,119,11,121,128,176,1896
660 DATA 200,24,239,62,48,119,43,24,236,62,14,205,48,252,205,60,1841
670 DATA 250,62,13,215,197,62,15,205,48,252,193,46,2,96,124,203,1983
680 DATA 31,203,31,201,31,203,31,230,15,205,189,251,215,124,230,15,2207
690 DATA 205,189,251,215,97,45,32,230,62,13,215,201,198,48,254,58,2313
700 DATA 216,198,7,201,62,16,205,48,252,17,85,255,6,4,213,197,1982
710 DATA 205,95,252,205,115,252,215,245,241,193,209,18,19,16,239,62,2581
720 DATA 13,215,62,17,205,48,252,221,33,85,255,17,0,16,33,0,1472
730 DATA 0,14,4,221,126,0,221,35,214,48,218,87,252,254,10,56,1760
740 DATA 2,214,7,254,16,210,87,252,71,254,0,40,3,25,16,1704
750 DATA 203,58,203,27,203,58,203,27,203,58,203,27,203,58,203,27,1964
760 DATA 13,32,208,229,193,205,43,45,205,227,45,62,13,215,201,203,2139
800 CLS : PRINT AT 5,5;" COMPILACAO COMPLETA. "
810 PRINT AT 7,2;"PREPARE O CASSETE PARA GRAVAR"
820 PRINT AT 9,4;"O NOME E TOOLKIT"" CODE"
830 SAVE "TOOLKIT"CODE 63489,2000
```

ROBÔS CONTROLADOS POR COMPUTADOR

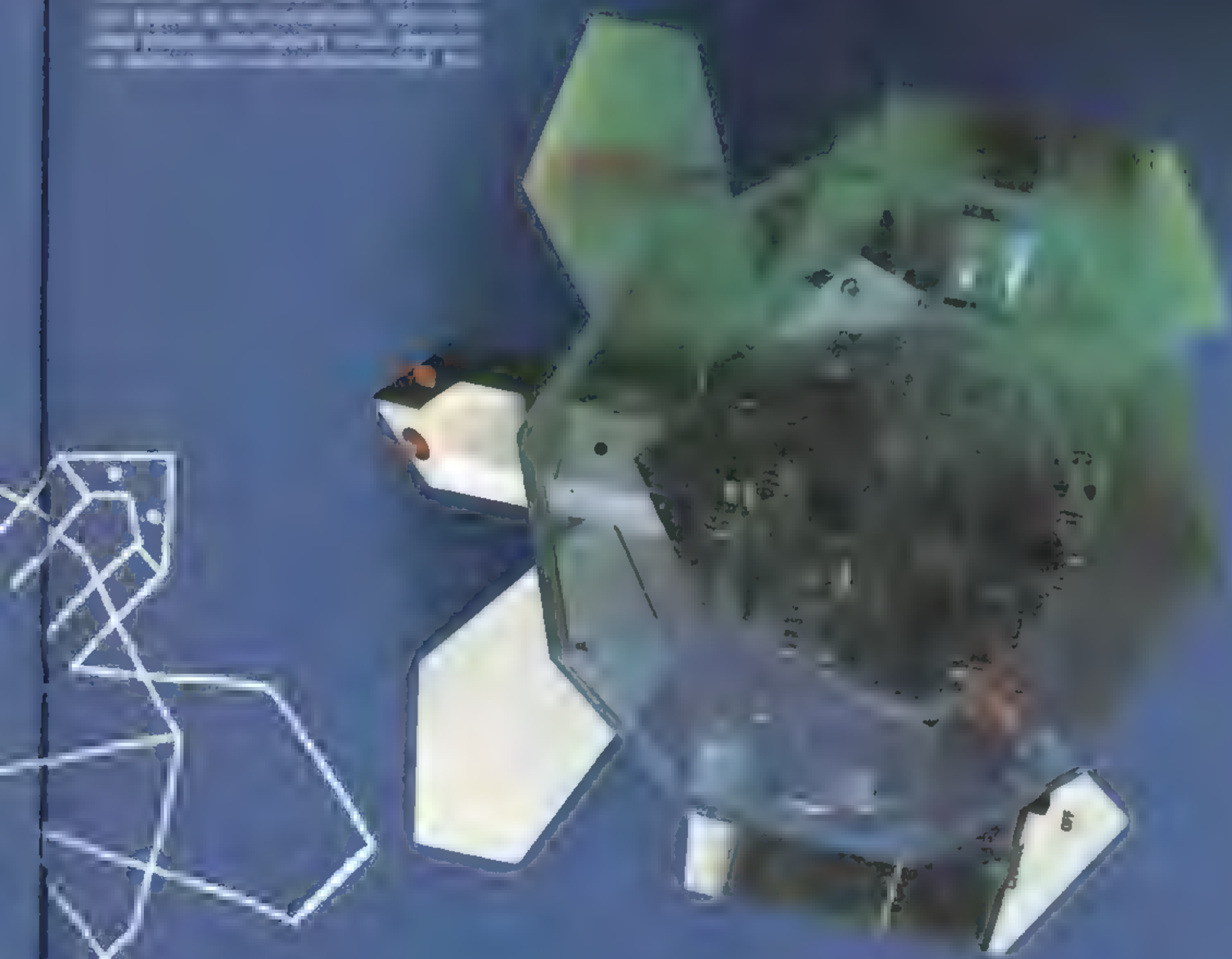
O QUE E UM ROBÔ?



[illegible]

[illegible] realizar tarefas simples [illegible] por exemplo, vigiar [illegible] apenas um con- [illegible] mecânicos e eletrô- [illegible] série limitada de

percepções e movimentos. Esses dispositivos separados são, em geral, controlados por um processador digital, montado no interior do próprio robô, ou em um microcomputador [illegible] fora dele.



COMUNICAÇÃO COM O EXTERIOR

Para se comunicar com um robô, os microcomputadores pessoais geralmente têm pelo menos uma porta de entrada/saída (E/S), através da qual podem entrar em contato com dispositivos mecânicos externos. Existem dois tipos de porta E/S: as seriais e as paralelas. Elas diferem entre si quanto à maneira de enviar e receber o fluxo de bits do periférico. Na maioria dos micros, a unidade básica de intercâmbio de informação é o byte — um conjunto de oito bits, ou dígitos binários. A informação armazenada em um byte é uma seqüência de oito algarismos 0 ou 1 — por exemplo: 00101101.

Na memória interna de trabalho do computador, dois bytes são rotulados especialmente para intercâmbio de dados com cada porta de entrada e saída. O primeiro byte, chamado de Registro de Direção dos Dados (*Data Direction Register*, DDR), determina o status da porta, definindo quando o intercâmbio será no sentido da entrada ou no sentido da saída. Em geral, se um bit no DDR tiver valor igual a 0, a porta correspondente está recebendo informações de fora; se for igual a 1, ela está transmitindo. Portanto, com o auxílio de um comando **POKE**, em BASIC, podemos ajustar cada um dos oito canais comandados pelo DDR, colocando um número decimal entre 0 (todos os canais recebendo) e 255 (todos os canais transmitindo, ou seja, DDR = 11111111 em binário). Colocando o valor 15, por exemplo, teremos DDR = 00001111 — isto é, os quatro primeiro canais estão emitindo, e os quatro últimos, recebendo. O endereço absoluto do DDR, a ser usado com o comando **POKE**, varia de computador para computador.

O segundo byte que controla a porta corresponde ao endereço da mesma. Um dado colocado nesse byte é convertido pela interface, conforme a aplicação a que se destina, em uma série de pulsos de voltagem, que são então passados para o dispositivo. No sentido oposto, as voltagens emitidas pelo dispositivo periférico são convertidas para bits pela interface, e colocadas no byte de endereço da porta. Dessa forma, o processador central — e qualquer programa introduzido nele — tem fácil acesso à porta de entrada e saída, não precisando se ocupar com as conversões a serem realizadas.

A linguagem de máquina, como algumas linguagens de alto nível, entre elas o BASIC, dispõe de instruções para escrever e ler numa determinada porta de saída. Essas instruções são denominadas **OUT** e **INP**, respectivamente. Em BASIC, por exemplo, um comando:

```
OUT 32,127
```

coloca o número decimal 127 na porta de saída número 32. Normalmente, a maioria dos micros de oito bits admite até 256 portas de E/S, numeradas de 0 a 255. Embora algumas delas estejam reservadas para periféricos já existentes, como o gravador cassete, as outras estão disponíveis para outros usos, como o controle de um robô.

O comando **INP** tem a mesma sintaxe e é usado para ler um byte no endereço da porta. Em geral, o computador não sabe quando este dado está disponível. Assim, costuma-se usar um outro comando em BASIC para testar a disponibilidade de um novo dado: **WAIT** (não confundir com o comando **PAUSE** dos micros da linha Sinclair).

Em uma porta paralela, os oito bits disponíveis no periférico ou byte de endereço da porta são transmitidos, simultaneamente, por oito fios distintos. Na porta serial, a transmissão se realiza em um bit de cada vez, em "fila indiana", por um único fio.

Quando chega a seu destino, no periférico, a informação é usada para controlar várias operações. Em um dispositivo eletromecânico, como um robô, ela é utilizada para ligar e desligar motores, relés etc. Após selecionar o endereço da porta de saída, o programador pode enviar sinais para destinos específicos. Por exemplo, se o valor é ajustado em 00110111, um sinal é mandado para as destinações correspondentes, através do terceiro, quarto, sexto, sétimo e oitavo condutores de uma porta paralela, ou, seqüencialmente, por um fio só, de uma porta serial.

ROBÔS PARA MICROCOMPUTADORES

Com o aumento do interesse pela robótica, surgiram nos últimos anos vários tipos de robôs baratos que podem ser controlados por um computador doméstico. As duas categorias básicas de robôs desse tipo são os braços robóticos e os robôs móveis (tartarugas e *buggies*), geralmente fornecidos em forma de kits para montar, e com o software necessário para operá-los.

Os braços robóticos procuram imitar os movimentos do braço humano. Costumam ter cinco pontos de movimento, ou *graus de liberdade*: ombro, cotovelo, pulso, garra (com ou sem rotação ao redor do pulso) e base do braço (movimento basculante). Braços robóticos podem ter até nove graus de liberdade. O nível de sofisticação do braço se reflete, é claro, em seu preço final.

Os braços mais baratos, como o Armatron, comercializado na Europa, não são programáveis. A designação de robô não se aplica a eles, portanto — não passam de um brinquedo mais elaborado. Outros, como o Armbot e o Armdroid 1, podem ser controlados por vários tipos de microcomputador. Têm cinco ou seis motores de passo variável e custam tanto quanto uma UCP de bom preço. Seus movimentos, realizados um por vez ou de modo contínuo, são controlados de modo direto ou por uma seqüência armazenada no micro. É possível adicionar pausas, alterar a velocidade do braço e editar as seqüências armazenadas, para modificar ou acrescentar novas fases de movimento. O programa para controle do braço pode ser escrito em Assembler ou BASIC, com chamadas às rotinas em código de máquina que controlam as portas de E/S. Estas são fornecidas pelo fabricante que também coloca à disposição do usuário listagens-exemplo de BASIC, com comentários e explicações.

Existem ainda braços robóticos que utilizam servomotores, em vez de motores de passo. Um servomotor possui um sensor de posição que verifica continuamente se as posições determinadas por comandos estão sendo atingidas. Circuitos integrados de preço bastante acessível (amplificadores lineares) permitem um controle razoável do periférico. Um exemplo de braços desse tipo é o Beasty, comercializado na Inglaterra para os micros da linha BBC.

O Beasty tem diversos implementos úteis para o desenvolvimento de projetos de robótica aplicada. Uma câmara miniaturizada, chamada Snap, permite que o micro "veja" e mova o braço. Esse dispositivo contém um circuito integrado sensível à luz, que transmite uma imagem digitalizada com uma resolução de 128 × 256. Pesa menos que 45 g, mede 8 × 10 cm, e é capaz de captar até vinte imagens por segundo. Vários programas interessantes são fornecidos com o Snap, possibilitando, entre outras coisas:

- exibir as imagens captadas no vídeo do microcomputador. A imagem pode ser "fixada", como se fosse uma foto, armazenada em disco ou fita, e, posteriormente, reproduzida por meio de uma impressora gráfica;

- animar uma seqüência de imagens (vinte quadros), como se fosse um filme ou desenho animado;

- comparar duas imagens tomadas em períodos distintos. Esse recurso tem larga aplicação, por exemplo, na monitorização de residências, edifícios ou lojas, no sentido de evitar a penetração de estranhos. Se a imagem tiver se modificado além de um limite prefixado, um alarme pode ser ativado. O programa mostra um gráfico com o número de mudanças, em função do tempo;

- reconhecer cenas e objetos e movimentar o braço em função disso.

Muitas empresas que fabricam jogos mecânicos de armar, como Meccano e Lego, lançaram kits com braços robóticos, que podem ser conectados a microcomputadores baratos, como o Sinclair Spectrum. Esses jogos são muito educativos — aprende-se muito sobre o funcionamento de um robô, quando se tem a oportunidade de armar e programar um sistema desde o início.

ROBÔS MÓVEIS

Ao contrário dos braços robóticos, de base estacionária, alguns robôs para microcomputadores, como as tartarugas e os *buggies*, movimentam-se sobre rodas. O primeiro robô desse tipo data dos anos 40, quando o cientista britânico Ross Ashby desenvolveu uma tartaruga eletrônica que tinha a tarefa de achar uma tomada para alimentar suas baterias. O "bichinho", apesar de muito simples, exibia notáveis e surpreendentes padrões "comportamentais".

A primeira tartaruga para micros apareceu no Laboratório de Inteligência Artificial do MIT (*Massachusetts Institute of Technology*), nos Estados Unidos. Era o "filhote" da equipe do Prof. Seymour Papert, criador da linguagem LOGO. Sua estrutura também é simples: consta de uma campânula hemisférica de acrílico, montada sobre uma base móvel com duas rodas independentes, acionadas por motores controlados pelo microcomputador. A campânula pode ser acoplada a um sensor de contato, que se comunica com o computador quando o robô colide com algum objeto ou parede. Na "barriga" da tartaruga, um solenóide controla um porta-caneta, que levanta ou abaixa sob controle do computador. Assim, a tartaruga pode ser "ensinada" a traçar desenhos sobre um papel colocado no chão.

As tartarugas têm sido muito utilizadas em escolas secundárias, para ensinar conceitos de programação a crianças menores. Empregando a linguagem LOGO ou uma simplificação da mesma (com comandos de uma tecla), a criança aprende a programar a tartaruga para realizar movimentos complexos. Um dos jogos preferidos consiste em dar comandos pelo teclado, até que a tartaruga entre em sua "casinha". Os comandos em LOGO, digitados em um microcomputador, são traduzidos por uma interface de software para códigos binários de controle das portas de saída. Esses sinais são enviados por um "cordão umbilical" ligado à tartaruga, ou, nos modelos mais sofisticados, como a *Valiant Turtle*, por meio de sinais infravermelhos. Uma placa com circuitos eletrônicos, na tartaruga, traduz os comandos binários em ações sobre os motores das rodas e da caneta. A tartaruga tem ainda um olho luminoso, que acende quando ela se movimenta.

Os comandos do LOGO permitem a movimentação da tartaruga segundo um sistema geométrico, em que a própria tartaruga é o ponto de referência. Examinaremos esses comandos em detalhe, na série de artigos sobre novas linguagens. Adiantamos aqui, entretanto, as instruções mais típicas:

FORWARD - Move a tartaruga, em linha reta, para a frente.
BACK - Move a tartaruga, em linha reta, para trás.
RIGHT - Vira a tartaruga para o lado direito.
LEFT - Vira a tartaruga para o lado esquerdo.
PENUP - Levanta a caneta, interrompendo o desenho.
PENDOWN - Abaixa a caneta, permitindo o desenho.
REPEAT - Repete uma seqüência de movimentos várias vezes.
TO...END - Armazena uma seqüência programada de movimentos, que recebe um nome (procedimento).

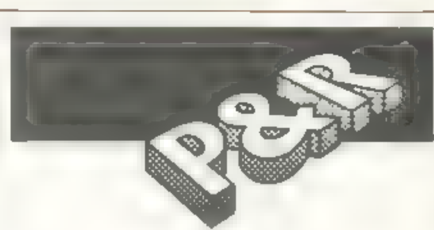
Para fazer a tartaruga desenhar um triângulo com lados de duzentos passos (unidades de deslocamento), as instruções a serem dadas são:

```
FORWARD 200
RIGHT 120
FORWARD 200
RIGHT 120
FORWARD 200
RIGHT 120
```

Poderíamos escrever o mesmo com mais economia:

```
REPEAT 3 [FORWARD 200 RIGHT
120]
```

Ou, ainda, armazenar as instruções na forma de um procedimento denominado **TRIANGULO**:



Há robôs para micros no Brasil?

Os leitores que se interessaram pelo assunto com certeza ficarão frustrados com a resposta: atualmente não há, no Brasil, robôs comerciais para conexão a microcomputadores.

A Universidade Estadual de Campinas (Unicamp) construiu uma tartaruga semelhante à criada pelo MIT, para uso em seus projetos com a linguagem LOGO. Porém, essa tartaruga não chegou a ser produzida industrialmente. Grupos de trabalho da Universidade de São Paulo (USP) e da Universidade Federal do Rio de Janeiro (UFRJ) desenvolveram protótipos de braços robóticos, como os descritos no artigo, mas estes também não são comercializados ainda.

Para quem tem condições, resta a possibilidade de mandar trazer do exterior um dos modelos de robô mais conhecidos, como a *Terrapin Turtle* ou a *Valiant Turtle* (tartarugas), o Hero I (robô *buggy* da Zenith/Heathkit), Armbot, Armdroid ou Beasty (braços robóticos) e kits da Lego.

```
TO TRIANGULO
REPEAT 3 [FORWARD 200 RIGHT
120]
END
```

Uma vez definida, a palavra **TRIANGULO** passa a fazer parte do vocabulário LOGO. Depois, para traçar a figura, basta digitar o comando **TRIANGULO** pelo teclado.

Pode-se utilizar um mesmo procedimento dentro de outros, destinados a realizar movimentos mais complexos. Por exemplo, para fazer com que a tartaruga desenhe uma flor formada de doze pétalas triangulares iguais, digitamos o seguinte:

```
TO FLOR
REPEAT 12 [TRIANGULO RIGHT 30]
END
```

O LOGO é uma linguagem muito versátil, também empregada para fazer cálculos matemáticos, manipular dados simbólicos como listas, palavras e nomes etc. Associado a uma tartaruga, tem o poder de tornar muito divertido o aprendizado da programação.

Programas em outras linguagens, como o BASIC, também podem ser usados para controlar a tartaruga.

A TORRE DE BABEL

Existem atualmente mais de duzentas linguagens de programação. Algumas delas prometem tornar-se tão conhecidas quanto o BASIC no decorrer dos próximos anos.

Até agora, focalizamos em *INPUT* duas linguagens de programação: Assembler e BASIC. Estas são, de fato, as linguagens mais difundidas entre os usuários dos microcomputadores pessoais e domésticos existentes atualmente no mercado brasileiro. Não são, porém, as únicas disponíveis.

Algumas linguagens vêm se tornando gradativamente mais conhecidas e utilizadas em micros pessoais — em particular o LOGO, o PASCAL e o FORTH. Outras, criadas já há algum tempo, mas destinadas a computadores de maior porte, prometem estender seu uso aos micros, podendo se tornar o futuro "esperanto" das máquinas de menor porte. Entre elas, incluem-se o LISP e o PROLOG. Entretanto, como as linguagens mais recentes ainda se encontram em processo de aceitação e difusão, nem sempre estão disponíveis para os micros de todas as marcas.

Na série que se inicia com este artigo, apresentaremos duas linguagens de programação cuja popularidade tem crescido dia a dia entre os usuários de micros: o LOGO e o PASCAL. Cada uma tem características que permitem sua aplicação seja como linguagem geral de programação (como o PASCAL), seja como linguagem destinada a áreas específicas — a área educativa, por exemplo (como o LOGO).

Examinaremos aqui alguns aspectos do desenvolvimento de linguagens de programação, a evolução histórica das linguagens existentes e os critérios usados em sua classificação.

Há atualmente mais de duzentas linguagens de programação catalogadas. À primeira vista, podemos ter a impressão de que se trata de uma enorme e bíblica "torre de Babel". No entanto, as linguagens de programação, tal como as linguagens naturais, podem ser enquadradas em linhas evolutivas, ou *famílias* de linguagens. Dentro de uma família, as semelhanças são maiores que as diferenças, o que facilita muito a compreensão e o aprendizado das linguagens que a compõem.

NOVAS LINGUAGENS

Um dos fatores condicionantes do aparecimento de novas linguagens de programação é o progresso tecnológico do hardware dos computadores — ou seja, a disponibilidade de memórias centrais cada vez maiores, o aumento na velocidade do processador central (UCP), o custo mais baixo das memórias auxiliares de disco, o aparecimento de redes de computadores etc.

Nos últimos anos, as linguagens de alto nível — que se parecem mais com a linguagem natural humana do que com a binária, entendida somente pelos computadores — sofreram grande evolução. Avanços técnicos recentes permitem que elas sejam utilizadas como a linguagem nativa dos computadores, no lugar da linguagem de máquina. Com isso, a programação se torna muito mais fácil, podendo ser realizada, em muitos casos, pelos próprios usuários.

No final da década de 70, o Assembler e o BASIC eram praticamente as únicas linguagens para micros de custo mais baixo, pois o tamanho das memórias ROM disponíveis limitavam muito a complexidade e a extensão do programa interpretador, que deveria ficar permanentemente na memória da máquina.

Entretanto, à medida que o espaço para a memória foi sendo ampliado, cresceu também a demanda por outros tipos de linguagem — o que era de se esperar, dado o número cada vez maior de usuários e de possibilidades de aplicação. Surgiram então as primeiras implementações práticas, em microcomputadores, de FORTH, PASCAL, LOGO, FORTRAN, PILOT, C, LISP, PROLOG, SMALLTALK e várias outras linguagens. Muitas delas só podiam ser usadas, anteriormente, em computadores de grande porte.

Mesmo algumas linguagens de maior complexidade, que encontram aplicações em áreas comerciais ou científicas mais "pesadas", como o COBOL, APL, PL/1, SNOBOL, MODULA, ALGOL e ADA, já podem ser utilizadas em microcomputadores da faixa profissional.

Ao que tudo indica, a difusão dessas linguagens tende a crescer, conforme os usuários forem se familiarizando com as suas vantagens. A expansão na capacidade e velocidade dos micros fornecerá as bases para esse avanço.

Outro fenômeno registrado na microinformática é a multiplicação de programas que funcionam, ao mesmo tempo, como um aplicativo (bancos de dados, planilhas eletrônicas etc.) e como linguagem de programação. É o caso do dBASE II (banco de dados) e do FRED (planilha eletrônica), entre outros.

O desenvolvimento de muitos desses novos aplicativos "genéricos" deu origem a linguagens altamente específicas, usadas para simulação (DYNAMO, GPSS), controle de equipamentos industriais (PEARL), controle de interfaces e equipamentos musicais (AMPLE) etc.

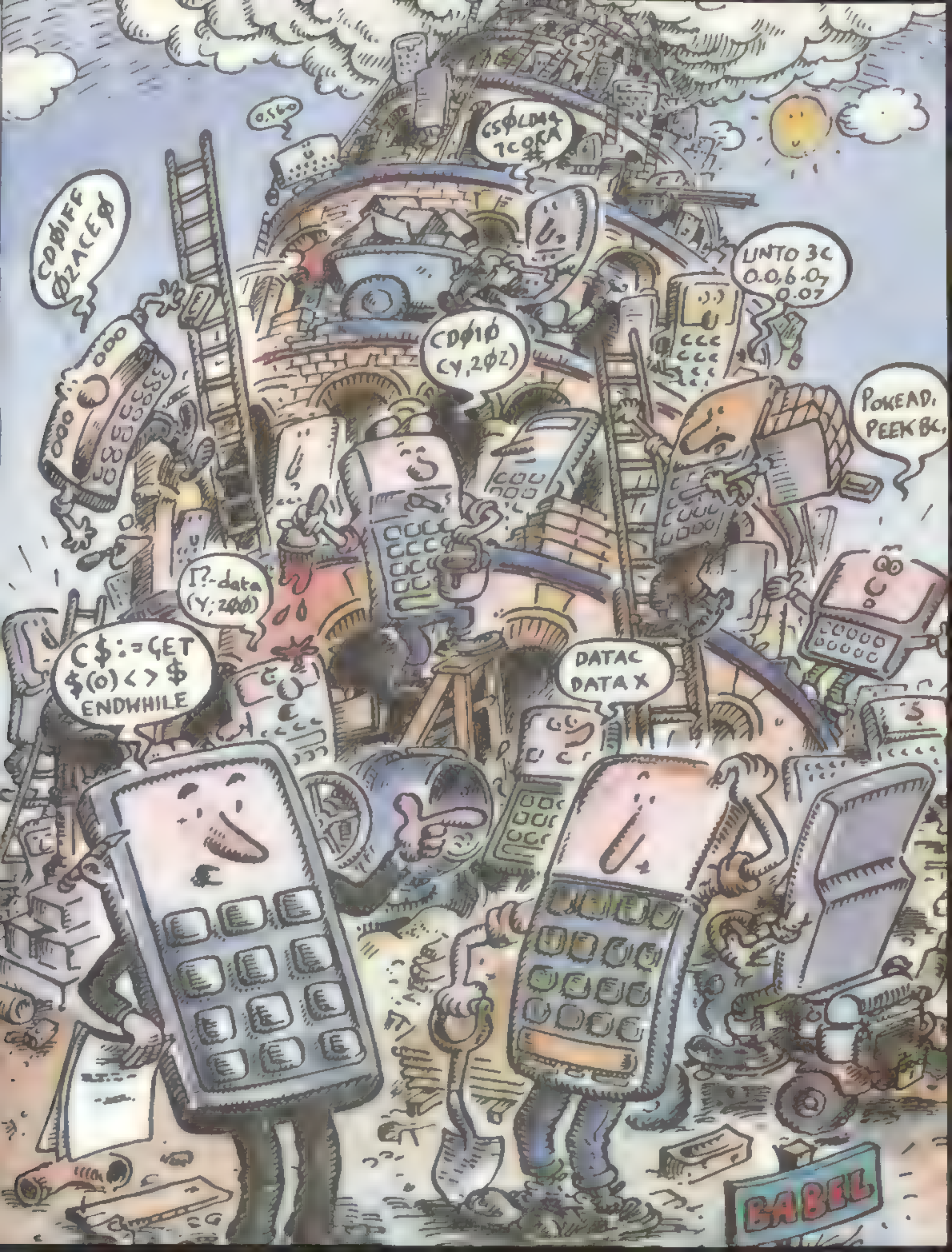
O DESENVOLVIMENTO DE LINGUAGENS

Nada impede que todas as linguagens de programação venham a ser desenvolvidas para microcomputadores. Mas parece pouco provável que isso ocorra, visto o grande número de linguagens existentes. Das mais de duzentas desenvolvidas até agora, não chegam a vinte as que se tornaram conhecidas e utilizadas.

Como as linguagens naturais, as linguagens de programação passam por contínuo processo de transformação, revisão e alteração. O FORTRAN (*FORmula TRANslation*), por exemplo, evoluiu tanto a partir da versão original, lançada em 1954, que, em alguns aspectos, é uma linguagem diversa.

Além desse contínuo aperfeiçoamento, as linguagens também passam por um processo de geração de uma enorme quantidade de "dialetos" ou variantes, desenvolvidas para implementação em linhas específicas de hardware, em diferentes sistemas e organizações.

Uma das limitações mais sérias encontradas pelo usuário de micros é justamente a incompatibilidade entre diferentes dialetos do BASIC. Um progra-



CDØIFF
ØZACEØ
CSØLDA4
7COFA
UNTO 3C
0.0,6.07
0.07
CDØ1Ø
(Y,2Ø2)
POKEAD.
PEEK BC.
I?-data
(Y;2ØØ)
C$:=GET
$(0)<>$
ENDWHILE
DATAC
DATA X
BABEL

ma desenvolvido segundo o BASIC da linha Apple, por exemplo, em geral não pode ser executado em um micro de outra linha, como o Spectrum ou o MSX.

Apesar de todo o esforço despendido na elaboração de um padrão internacional, o ANSI BASIC, o problema ainda não foi solucionado. A inadequação desse padrão é evidente: tanto que não há uma única marca de computador que o adote integralmente. Por isso, o interpretador BASIC, criado pela firma norte-americana Microsoft, tem sido usado como uma espécie de padrão por fabricantes de máquinas tão distintas como o Apple, o TRS-80 e o MSX.

O projeto do MSX, aliás, é fruto de uma das últimas tentativas de se padronizar hardware e software de microcomputadores pessoais através da implementação de mais um dialeto do BASIC Microsoft. Esse padrão tem tido excelente repercussão mundial, principalmente no Japão (seu país de origem), na Europa, e no Brasil.

A história das outras linguagens não foi muito diferente: o sucesso de um padrão resultou sempre da conquista de uma boa fatia do mercado pela empresa que o desenvolveu. O FORTRAN, por exemplo, foi desenvolvido pela IBM e, como era distribuído gratuitamente com os computadores dessa empresa, passou a ser amplamente adotado e seguido, inclusive por outros fabricantes. Contando com uma grande base de usuários, a própria IBM pôde realizar melhorias subseqüentes do FORTRAN — como o FORTRAN II, o FORTRAN IV, as versões G e H, o FORTRAN 77 etc.

Entretanto, uma linguagem como o PASCAL, que talvez seja a segunda linguagem mais popular para micros, possui pelo menos sete versões diferentes. Cada uma delas é um semipadrão para o PASCAL relativo ao sistema operacional sob o qual roda.

AS RAÍZES DA LINGUAGEM

A torre de Babel é uma conseqüência inevitável da necessidade de desenvolvimento de novas linguagens.

Voltemos ao exemplo do FORTRAN. Seguramente, foi a primeira linguagem de alto nível desenvolvida. Sua criação atendeu a duas exigências: a resolução de fórmulas matemáticas e científicas, e o uso eficiente dos recursos computacionais disponíveis. Nesses aspectos, foi um sucesso. Mas, em dois outros pontos, que não tinham sido julgados importantes pelos projetistas do FORTRAN, a linguagem mostrou-se deficiente. Ela é excelente para fins matemáticos e científicos, mas extremamente inadequada (ou mesmo inútil) para tarefas como processamento de textos ou bancos de dados. Além disso, não é uma linguagem estruturada, o que restringe a ação do programador.

Na década de 60, desenvolveu-se uma outra linguagem, o ALGOL 60, com uma filosofia distinta de projeto. Seus pontos fracos e fortes diferem bastante daqueles que caracterizam o FORTRAN — sobretudo por se tratar da primeira linguagem de programação estruturada. Ela foi projetada para ser essencialmente algorítmica (daí seu nome, *ALGOrithmic Language*), ou seja, para corresponder não às fórmulas matemáticas, mas ao processo inerente usado na resolução de um problema (algoritmo). A unidade básica do ALGOL é o *procedimento*, um bloco autônomo de código, com uma tarefa claramente definida, que trabalha com tipos predefinidos de variáveis (*declarações*). Por essa razão, todas as linguagens derivadas diretamente do ALGOL, como o PASCAL, o MODULA-2 e o ADA, são classificadas como *procedimentais, declarativas, algorítmicas* e *estruturadas*.

Embora não tenha sido muito usado, o ALGOL 60 desempenhou um papel de destaque, gerando um grande número de linguagens. Já o COBOL, considerado o terceiro "avô" da computação, foi intensamente utilizado — mas só deixou um "descendente", o PL/1.

O COBOL (*Common Business — Oriented Language*) surgiu como resposta à conclusão — enunciada pelo Ministério da Defesa dos Estados Unidos — de que as instalações militares e centros de pesquisa necessitavam de uma linguagem de programação comum, se quisessem operar o sistema americano de defesa como um todo, efetivamente. Desenvolvida a partir de um esquema de cooperação entre governo e indústria, essa linguagem difundiu-se rápida e amplamente. É provável que seja, hoje, a linguagem mais utilizada para aplicações comerciais em computadores de grande porte. Uma das razões que explicam a grande aceitação do COBOL é sua proximidade do inglês natural.

EVOLUÇÃO

As linguagens de alto nível anteriormente mencionadas deram origem à grande maioria das linguagens de programação da década de 60, muitas das quais ainda em uso. O BASIC, por exemplo, é um descendente direto do FORTRAN. As diferenças entre ambos refletem o intuito de se chegar a uma linguagem de fácil uso pelos iniciantes em programação. O ALGOL 60 deu origem ao ALGOL 68 — tão diferente, que pode ser considerado uma nova linguagem, apesar do mesmo nome —, bem como ao PASCAL e ao SIMULA. De uma combinação entre o ALGOL e o COBOL surgiu o PL/1. Do PASCAL e do SIMULA nasceu um grupo de linguagens procedimentais e algorítmicas, que incluem o MODULA, o MESA, o EUCLID e o ADA. Finalmente, uma linguagem bastante poderosa, voltada à programação de objetos, originou-se do SIMULA — o SMALLTALK.

LINGUAGENS IMPERATIVAS

Todas as linguagens citadas são classificadas como *imperativas*, pois operam através de uma seqüência de comandos declarativos — isto é, comandos que determinam um tipo altamente específico de ação sobre dados da memória, rotulados na forma de variáveis. Muitas pessoas estão tão acostumadas a essa forma de programação, que imaginam tratar-se da única possível. Entretanto, outras famílias, que incluem linguagens como o FORTH, o LISP e o LOGO, têm estruturas de dados e de processamento bastante diferentes. Elas constituem o grupo de linguagens chamadas *funcionais* ou *aplicativas*, operando basicamente através da aplicação recursiva de funções.

LINGUAGENS FUNCIONAIS

As linguagens funcionais, como o próprio nome sugere, atuam sobre os dados por meio de funções ou procedimentos. Um procedimento é algo bem mais poderoso do que um comando. Vejamos por quê: toda linguagem de programação tem um vocabulário fixo de instruções, chamado *conjunto de instruções*. As linguagens imperativas utilizam um conjunto *fixo* de instruções, ou seja, um conjunto que não pode ser ampliado pelo próprio usuário. A linguagem funcional, ao contrário, não faz distinção entre os comandos, declarações ou funções pertencentes ao conjunto original e as funções ou procedimentos criados pelo usuário. Por isso, diz-se que certas classes de linguagens funcionais são *extensíveis*.

O processamento baseado em funções apresenta ainda outra vantagem: ela age sobre parâmetros de entrada e devolve parâmetros de saída, sem "saber" necessariamente o que eles representam. Isso justifica a classificação das funções como *generalizáveis*.

As linguagens funcionais podem ser divididas em módulos (blocos autônomos de processamento) mais facilmente do que as imperativas, e possuem uma estrutura interna hierárquica que é usada tanto para o armazenamento do programa quanto dos dados. No LISP ou no LOGO, por exemplo, uma lista definida como uma estrutura de dados pode ser usada para armazenar um programa. Existem comandos para modificar e executar um programa, armazenado em uma lista (**EXECUTE**, em LOGO, e **EVAL**, em LISP). Essa propriedade de uma linguagem é chamada *indireção*.

Já em uma linguagem imperativa, como o BASIC, programa e dados são elementos inteiramente distintos. O conjunto de instruções não pode ser ampliado e a ordem de execução é muito importante: se ela sofrer alguma modificação, determinados valores de memória não serão criados ou atualizados.

Linguagens como o LISP permitem uma abordagem bem diferente. Em um certo sentido, cada programa em LISP — ou nas linguagens que dele derivam — é uma extensão da linguagem original, com o acréscimo das novas funções definidas pelo usuário. Com uma linguagem funcional, o programador tem maior controle sobre o que faz do que com uma linguagem imperativa.

As três propriedades mencionadas — recursão, extensibilidade e indireção — são fundamentais em um campo muito atual das ciências da computação: a Inteligência Artificial ("raciocínio" e "aprendizado" por máquina). LISP e LOGO, aliás, foram criados especificamente para facilitar o processamento simbólico, não-numérico, característico das aplicações de Inteligência Artificial. Ambos nasceram no Laboratório de Inteligência Artificial do MIT (Massachusetts Institute of Technology, nos EUA).

EXTENSÕES

Como já dissemos, as linguagens de computador, da mesma forma que as linguagens naturais, não permanecem estáticas: elas evoluem constantemente, absorvendo aspectos de outras linguagens. Com isso, as diferenças entre linguagens imperativas e linguagens funcionais tendem a diminuir.

Tomemos como exemplo o BASIC: os dialetos mais difundidos do Microsoft BASIC (para as linhas Apple, TRS-80, TRS-Color e MSX) são linguagens puramente imperativas. Seus descendentes mais modernos, porém, como o QuickBASIC (da própria Microsoft), possuem características funcionais, entre elas a programação estruturada.

Um processo semelhante ocorreu com o BASIC original das linhas Sinclair. O SuperBASIC do modelo QL Spectrum, que dele deriva, tem não só estruturas — como **CASE SELECT** — próprias do PASCAL, como também oferece a possibilidade de se chamar funções apenas pelo nome e de se utilizar a recursão. Desenvolvimentos futuros da linguagem BASIC provavelmente a afastarão ainda mais do padrão original (ANSI), incorporando elementos de linguagens funcionais como o FORTH e o LISP.

Como já vimos, o LISP foi desenvolvido como uma linguagem para o campo de Inteligência Artificial, e, ainda hoje, predomina nessa área. Uma de suas principais características é o intenso uso de cadeias de caracteres (*string*). Outra linguagem útil na mesma área é o SNOBOL (*StriNg Oriented SymBOlic Language*), originalmente desenvolvida para o processamento de linguagem natural em Inteligência Artificial, Linguística etc., em tarefas como a determinação da autoria de manuscritos, análise estilística de textos literários e geração automática de poemas.

Também merece destaque a recente evolução das *linguagens de desenvolvimento de sistemas*. São linguagens de alto nível, mas que incorporam recursos avançados de trabalho direto com a memória, processador e periféricos do computador. Entre as linguagens desse tipo, a de maior sucesso é o C, que foi usada para desenvolver o sistema operacional UNIX. Ela possui um poderoso conjunto de ferramentas de desenvolvimento de software e tem sido utilizada sobretudo por programadores mais experientes. Para o usuário comum, o C parece um tanto inacessível — está entre uma linguagem de alto nível, como o PASCAL, e um Assembler, bem mais flexível, mas difícil de usar. Mesmo assim, essa linguagem vem se difundindo, principalmente entre os amadores que desejam dominar sua máquina sem ter o trabalho de aprender Assembler. Existem diversas implementações do C para micros, inclusive o Spectrum.

O SmallTalk é outra linguagem que segue de perto a filosofia do LISP. Típica de uma nova família, a das *linguagens de programação de objetos*, ela constitui um "ambiente" completo de programação, como o C, e não apenas uma linguagem. O SmallTalk tem, no entanto, diversos recursos voltados para a máxima simplificação do conceito de programa e da tarefa de programação. Nessa linguagem, cada item de um programa é considerado um objeto. A programação é feita através do intercâmbio de "mensagens" entre objetos. Como não existe outro tipo de processamento, qualquer ação em um programa é sempre a mesma coisa: uma mensagem enviada para um objeto, que, por sua vez, envia mensagens para outros objetos.

Os objetos podem realizar processamentos internos, conforme a mensagem que recebem. Um objeto do SmallTalk é, assim, semelhante a uma função LISP ou a um procedimento PASCAL. Só que, em vez de construir procedimentos e incorporá-los a uma estrutura em árvore, como no LISP e LOGO, os objetos podem ser considerados entidades independentes — como os registros de uma base de dados, em que cada registro interroga ou responde aos demais. Em outras palavras, um objeto pode conter tanto dados quanto programas.

A idéia não é muito fácil de entender, a não ser que se observe o funcionamento do sistema. Infelizmente, ainda não se encontram implementações de SmallTalk para todos os micros. Mas certamente vale a pena investigar essa linguagem, pois ela representa uma tendência importante para o futuro. Sistemas operacionais de micros modernos, como o MacIntosh, da Apple, foram muito influenciados pela filosofia SmallTalk, inclusive quanto ao tratamento da interface com o usuário, que é feita através de "janelas" múltiplas, menus e acionamento do "mouse".

LINGUAGENS DE QUARTA GERAÇÃO

Certas linguagens aplicativas mais recentes (de quarta geração) foram desenvolvidas a fim de dotar o programador de poderosíssimos recursos, denominados *macros*. Com efeito, nessas linguagens, um único comando ou função equivale a centenas ou milhares de instruções de uma linguagem imperativa comum, como o BASIC. Um exemplo de linguagem de quarta geração para micros, o dBASE II (e sua extensão para micros de dezesseis bits, o dBASE III), é, ao mesmo tempo, um sistema aplicativo – mais especificamente, um sistema gerenciador de bases de dados, ou SGBD — e uma linguagem de programação.

Outra linguagem de quarta geração em franca ascensão é o PROLOG (*PROgramming in LOGic*), desenvolvido na França e adotado pelos japoneses em seu programa de computadores de quinta geração. O PROLOG está para o campo da Inteligência Artificial como o dBASE para a área de gerenciamento de dados. Existem PROLOG para micros de oito e dezesseis bits.

AVALANCHE: LISTAGEM COMPLETA

O jogo já foi todo digitado. Willie está pronto para escalar a montanha, enfrentar o mar, as pedras, as cobras e os buracos. Será que o programa vai funcionar?

Avalanche agora está completo, mas pode precisar de alguns acertos. É pouco provável que você tenha digitado um programa tão longo sem cometer alguns enganos. Dedique-se à resolução de eventuais problemas causados ao criar ou copiar erros. Para descobri-los, verifique suas listagens Assembly; se achar conveniente, remonte-as.

Depois que as rotinas separadas estiverem funcionando, carregue-as na memória, salve-as numa fita e teste o conjunto completo, chamando-o com o comando de execução de código de máquina de seu microcomputador.

Por sua estrutura modular, o programa do videogame *Avalanche* é fácil de corrigir. Caso encontre algum problema durante a execução do jogo, tenha o cuidado de identificar a parte que apresentou o defeito e concentre nela seus esforços para descobrir o erro.

Mesmo que todas as rotinas funcionem adequadamente quando rodadas uma a uma, é possível que o programa completo apresente falhas. Rotinas chamam outras rotinas — e se uma delas contiver algum tipo de erro ou estiver montada em lugar inadequado, o programa poderá não funcionar. Estamos, por isso, publicando uma listagem hexadecimal completa do jogo. Utilize-a para checar sua versão final.

Depois de montar *Avalanche*, o programa estará na memória do computador como uma série de números hexadecimais em posições sucessivas, tal qual mostramos neste artigo. O número de quatro dígitos na coluna esquerda corresponde ao endereço da posição de memória ocupado pelos dois primeiros bytes de programa naquela linha. Os pares subseqüentes de dígitos ocupam as posições seguintes. Você tem, portanto, a possibilidade de examinar cada byte de *Avalanche* através do conteúdo da posição de memória correspondente e verificar se cada instrução do jogo foi montada corretamente.

Ao contrário de alguns outros jogos de computador, *Avalanche* nunca chega a entediar, pois seus parâmetros de execução podem ser alterados. A construção modular do programa deixa o usuário livre para fazer as modificações que quiser, permitindo-lhe adaptar *Avalanche* às suas preferências pessoais. Seja curioso. Experimente. Afinal, o jogo agora é seu. E você pode fazer dele o que bem entender.

O jogo está escrito na região 50000 da memória para que possa ser montado pelo Assembler de *INPUT*.

Se você não está usando o Assembler de *INPUT*, talvez precise deslocar o jogo na memória por causa do espaço ocupado pelo próprio Assembler. A melhor alternativa é a região 20000. Para fazer a transferência basta mudar para 2 os endereços que iniciam com 5.

Lembre-se de verificar se a rotina que executa a música não apagou outras rotinas ao ser deslocada.

Use o monitor de código de máquina para analisar a memória do seu micro e ver se ela confere com a listagem em hexa dada a seguir. Se você tiver deslocado o jogo, as posições de memória serão diferentes, mas a sequência de códigos hexa deve ser a mesma.

```
DEA8 18 3C 3C 18 3C 3C 3C 3C
DEB0 3C 3C 18 18 18 18 18 1E
DEB8 01 03 03 01 00 01 01 01
DEC0 80 C0 60 80 00 00 00 E0
DEC8 0E 00 01 02 04 08 04 00
DED0 00 00 80 40 20 20 30 00
DED8 00 00 00 00 18 3C 3C 18
DEE0 00 10 10 1E E0 00 0C 24
DEE8 42 82 43 00 00 00 00 00
DEF0 00 00 00 00 01 03 03 01
DEF8 00 00 00 00 80 C0 C0 80
DF00 00 01 01 01 0E 00 01 02
DF08 00 00 00 E0 00 00 80 40
DF10 04 08 04 00 00 00 00 00
DF18 20 20 30 00 00 00 00 00
DF20 1C 3E 7F FF FF FE FC 38
DF28 03 07 0F 0F 0F 07 03 01
DF30 80 C0 E0 F0 F0 F0 E0 C0
DF38 00 00 07 18 20 40 40 80
DF40 00 00 1F A0 C0 00 00 00
DF48 00 00 80 40 5C 22 02 02
DF50 80 40 7C 02 02 01 00 00
DF58 00 00 00 04 0A 11 60 00
DF60 02 04 08 04 04 04 F8 00
DF68 00 00 78 86 01 01 00 00
DF70 00 00 1E 61 80 80 00 00
DF78 00 00 00 00 87 79 00 00
DF80 00 00 00 00 E1 9E 00 00
DF88 22 14 08 08 08 08 08 08
DF90 18 3C 36 3E 7E 3C 18 18
DF98 18 18 0C 0C 06 06 03 03
DFA0 06 06 0C 0C 18 18 30 30
DFA8 60 60 C6 C3 66 6C 38 38
DFB0 84 D6 FF FF FF FF FF FF
DFB8 00 01 03 07 1F 3F 7F FF
DFC0 00 00 FF FF 3C 3C FF FF
DFC8 06 08 76 FF FF FF 3E 3C
DFD0 10 10 10 38 38 38 38 38
DFD8 10 1C 7C 38 38 38 10 10
DFE0 00 00 00 20 51 8A 04 00
DFE8 00 00 00 82 45 28 10 00
DFF0 00 00 00 00 00 00 00 00
DFF8 00 00 00 00 00 00 00 00

E000 00 00 00 00 00 00 00 00
E008 00 00 00 00 00 00 00 62
E010 00 B4 05 E9 00 C7 04 83
E018 00 3E 04 DC 00 C4 03 4E
E020 00 BC 03 93 00 C4 03 05
E028 01 3E 04 6E 00 11 05 83
E030 00 6A 06 31 00 B4 05 6E
E038 00 11 05 E9 00 C7 04 62
E040 00 B4 05 93 00 B4 05 2C
E048 00 6A 06 62 00 B4 05 DC
E050 00 11 05 5C 00 0C 06 DC
E058 00 A7 07 43 4C 49 46 46
E060 48 41 4E 47 45 52 43 52
E068 45 41 54 45 44 20 42 59
E070 20 41 2E 44 4F 45 57 52
E078 49 54 54 45 4E 20 42 59
E080 20 50 2E 43 4C 41 52 4B
E088 20 41 66 74 65 72 20 61
E090 20 73 68 6F 72 74 20 77
E098 61 6C 6B 20 57 69 6C 6C
E0A0 69 65 20 20 20 20 20 20
E0A8 72 65 74 75 72 6E 73 20
E0B0 74 6F 20 66 69 6E 64 20
E0B8 74 68 65 20 67 6F 61 74
E0C0 73 20 68 61 76 65 20 20
E0C8 73 70 72 65 61 64 29 68
E0D0 69 73 20 70 69 63 6E 69
E0D8 63 20 67 6F 6F 64 69 65
E0E0 73 20 61 6C 6C 20 20 20
E0E8 6F 76 65 72 20 61 20 72
E0F0 6F 63 68 79 20 65 6D 62
E0F8 61 6E 6B 6D 65 6E 74 20
E100 20 20 20 20 20 20 20 20
E108 57 69 6C 6C 69 65 20 63
E110 65 74 73 20 6F 66 66 20
E118 74 6F 20 72 65 63 6C 61
E120 69 6D 20 68 69 73 20 20
E128 6C 6F 73 74 20 70 6F 73
E130 73 65 73 73 69 6F 6E 73
E138 2C 62 75 74 20 69 73 20
E140 68 61 6D 70 65 72 65 64
E148 62 79 20 66 61 6C 6C 69
E150 6E 67 20 62 6F 75 6C 64
E158 65 72 73 2C 70 6F 74 20
E160 68 6F 6C 65 73 20 20 20
E168 61 6E 64 20 76 69 63 69
E170 6F 75 73 20 73 6E 61 6B
E178 65 73 2E 54 6F 20 6D 61
E180 6B 65 20 20 20 20 20 20
```

- COMO DESCOBRIR OS ERROS
- TESTE O CONJUNTO COMPLETO
- VERIFICAÇÃO DA VERSÃO FINAL
- O CÓDIGO HEXADECIMAL
- ALTERAÇÕES POSSÍVEIS
- DESLOCAMENTO DO JOGO NA MEMÓRIA

```
E188 6D 61 74 74 65 72 73 20
E190 77 6F 72 73 65 20 74 68
E198 65 20 74 69 64 65 20 69
E1A0 73 20 72 69 73 69 6E 67
E1A8 61 6E 64 20 68 65 20 69
E1B0 73 20 69 6E 20 64 61 6E
E1B8 67 65 72 20 6F 66 20 62
E1C0 65 69 6E 67 20 63 75 74
E1C8 6F 66 66 2E 54 6F 20 68
E1D0 65 6C 70 20 57 69 6C 6C
E1D8 69 65 20 69 6E 20 68 69
E1E0 73 20 71 75 65 73 74 20
```

```
E1E8 72 65 61 64 20 74 68 65
E1F0 20 66 6F 6C 6C 6F 77 69
E1F8 6E 67 20 61 6E 64 20 70
E200 72 65 73 73 20 27 53 27
E208 74 6F 20 73 74 61 72 74
E210 2E 20 20 20 20 20 20 20
E218 20 20 20 20 20 20 20 20
E220 20 4E 20 20 20 20 2D 20
E228 52 75 6E 20 20 20 20 20
E230 20 20 20 20 20 20 20 20
E238 20 20 20 20 20 20 20 20
E240 20 4D 20 20 20 20 2D 20
```

```
E248 56 65 72 74 69 63 61 6C
E250 20 6A 75 6D 70 20 20 20
E258 20 20 20 20 20 20 20 20
E260 20 42 6F 74 68 20 2D 20
E268 44 69 61 67 6F 6E 61 6C
E270 20 6A 75 6D 70 53 43 4F
E278 52 45 2D 30 30 30 30 30
E280 30 4C 49 56 45 53 2D 35
E288 47 41 4D 45 20 4F 56 45
E290 52 20 21 21 21 23 23 21
E298 23 23 23 21 23 23 21 23
E2A0 21 23 23 23 23 21 23 21
```



```
E2A8 23 23 23 23 21 23 23 21
E2B0 23 23 23 CD 50 E3 3E 02
E2B8 D3 FE 3E 10 32 48 5C DD
E2C0 21 58 E0 06 05 3E 46 21
[illegible] 86 00 CD 2B E3 06 06 21
E2D0 CC 00 CD 2B E3 06 10 3E
E2D8 07 21 62 02 CD 2B E3 06
E2E0 12 21 A2 02 CD 2B E3 06
[illegible] 02 21 E8 FD 11 00 00 2B
E2F0 E5 ED 52 E1 20 F9 10 F1
E2F8 CD 50 E3 DD 21 88 E0 21
E300 20 00 3E 07 06 FF CD 2B
E308 E3 06 8A CD 2B E3 11 26
E310 00 19 06 64 3E 46 CD 2B
E318 E3 3E FD DB FE CB 4F 20
E320 F8 C9 3E FD DB FE CB 4F
E328 20 F8 C9 C5 F5 DD 7E 00
E330 CD 3E E3 F1 CD 69 E3 23
E338 DD 23 C1 10 EE C9 E5 21
E340 F8 3C 11 08 00 06 1F 90
E348 19 3D 20 FC E5 C1 E1 C9
E350 DD 21 00 40 21 00 1B 3E
E358 00 DD 77 00 DD 23 2B E5
E360 11 00 00 ED 52 E1 20 F1
E368 C9 F5 E5 C5 E5 D1 7A FE
E370 01 38 11 D5 11 00 07 19
E378 D1 7A FE 01 28 06 D5 11
E380 00 97 19 D1 D5 11 00 40
E388 19 D1 3E 08 C1 F5 0A 77
E390 24 03 F1 3D 28 03 F5 18
E398 F5 E1 F1 D5 11 00 58 19
E3A0 D1 77 D5 E1 C9 00 00 00
E3A8 00 00 00 00 00 00 00 00
E3B0 00 00 00 00 00 00 00 00
[illegible] 00 00 00 00 00 00 00 3E
E3C0 10 32 F0 DF DD 21 B2 E2
E3C8 06 20 C5 DC 35 E4 3E 00
E3D0 32 F1 DF DD 7E 00 DD 2B
E3D8 FE 21 20 0D 05 3A F0 DF
E3E0 3D 32 F0 DF 3E 01 32 F1
E3E8 DF 3A F0 DF 47 21 1F 00
E3F0 3E 2D CD 48 E4 01 B0 DF
E3F8 3A F1 DF FE 01 20 03 01
E400 B8 DF 3E 2C CD 69 E3 3A
E408 F0 DF 47 3E 17 90 47 3E
E410 20 11 20 00 19 CD 48 E4
E418 C1 10 AF 21 31 00 06 0C
E420 3E 29 DD 21 75 E2 CD 2B
E428 E3 21 71 00 06 07 CD 2B
E430 E3 CD 57 E4 C9 21 00 40
E438 06 D8 0E 1F 23 7E 2B 77
E440 23 0D 20 F8 23 10 F3 C9
E448 C5 01 00 3D CD 69 E3 11
E450 20 00 19 C1 10 F2 C9 3A
E458 00 E0 21 88 DF 47 04 11
E460 08 00 19 10 FD E5 C1 21
E468 BF 00 3E 3A CD 69 E3 3A
E470 00 E0 FE 00 28 0C F5 CD
E478 83 E4 F1 FE 01 28 03 CD
E480 A9 E4 C9 21 C9 01 CD 96
E488 E4 21 91 01 CD 96 E4 21
E490 3A 01 CD 96 E4 C9 06 04
E498 C5 01 00 3D 3E 2D CD 69
E4A0 E3 11 20 00 19 C1 10 F0
E4A8 C9 21 C9 01 CD BC E4 21
E4B0 91 01 CD BC E4 21 3A 01
E4B8 CD BC E4 C9 3E 04 01 90
E4C0 DF F5 3E 2B CD 69 E3 11
E4C8 20 00 19 F1 3D 20 F2 C9
E4D0 CD B3 E2 3E 05 32 FF DF
E4D8 3E 00 32 00 E0 21 00 00
E4E0 22 F9 DF 22 FB DF 22 FD
E4E8 DF 3E 13 32 65 EA 3E 06
E4F0 32 09 E0 21 E0 02 22 0A
E4F8 E0 21 82 00 22 01 E0 3E
E500 03 32 03 E0 3E 00 32 04
E508 E0 3E 02 32 05 E0 21 C1
E510 01 22 F4 DF 21 00 00 22
E518 F6 DF 3E 00 32 F8 DF 21
E520 DF 00 22 0C E0 3E 00 06
E528 05 32 06 E0 80 32 07 E0
E530 80 32 08 E0 CD BF E3 CD
E538 3B E6 21 77 00 3A FF DF
E540 06 30 80 CD 3E E3 3E 29
E548 CD 69 E3 CD 60 EA CD 11
E550 E7 CD 71 E6 CD AF E9 CD
E558 02 E6 CD AB E5 CD 7F E5
E560 3A FB DF EF 01 CA BC E9
E568 FE 02 CA 04 E9 06 32 3E
E570 FF 3D 20 FD 10 FB 3E FE
E578 DB FE CB 47 20 D0 C9 3A
E580 05 E0 3C CB 9F 32 05 E0
E588 01 68 DF FE 04 38 03 01
E590 78 DF 3E 2E 21 2A 00 C5
E598 CD 69 E3 23 CD 69 E3 C1
E5A0 21 44 00 CD 69 E3 23 CD
E5A8 69 E3 C9 3A 03 E0 3D 32
E5B0 03 E0 FE 00 28 01 C9 3E
E5B8 06 32 03 E0 3E 2D 01 00
E5C0 40 2A 01 E0 16 03 1E 02
E5C8 CD 5A E6 3A 04 E0 FE 00
E5D0 28 03 2B 18 01 23 22 01
E5D8 E0 01 38 DF 3E 2F 16 03
E5E0 1E 02 CD 5A E6 11 81 00
E5E8 ED 52 20 06 3E 00 32 04
E5F0 E0 C9 11 90 00 2A 01 E0
E5F8 ED 52 20 05 3E 01 32 04
E600 E0 C9 01 E0 DF 3A 09 E0
E608 CB 57 28 03 01 E8 DF 2A
E610 0A E0 3E 0F 16 20 D5 C5
E618 CD 69 E3 23 C1 D1 15 20
E620 F5 3A 09 E0 3D 32 09 E0
E628 20 10 3E 0A 32 09 E0 2A
E630 0A E0 11 20 00 ED 52 22
E638 0A E0 C9 21 37 00 DD 21
E640 F9 D7 06 06 C5 DD 7E 00
E648 06 30 80 CD 3E E3 3E 29
E650 CD 69 E3 23 DD 23 C1 10
E658 EB C9 E5 D5 E5 D5 CD 69
E660 E3 23 D1 15 20 F7 E1 11
E668 20 00 19 D1 1D 20 EC E1
E670 C9 3A 00 E0 FE 00 28 05
E678 FE 03 28 01 C9 3A 0E E0
E680 FE 01 28 55 2A 0C E0 01
E688 20 DF 3E 2A CD 69 E3 23
E690 3E 2D 01 00 3D CD 69 E3
E698 2A 0C E0 11 E0 01 ED 52
E6A0 28 5D 2A 0C E0 11 20 58
E6A8 19 7E FE 0F 28 51 FE 2D
E6B0 20 1A 2A 0C E0 01 00 3D
E6B8 3E 2D CD 69 E3 11 20 00
E6C0 19 22 0C E0 01 20 DF 3E
E6C8 2A CD 69 E3 2A 0C E0 2B
E6D0 22 0C E0 3E 01 32 0E E0
E6D8 C9 2A 0C E0 11 00 58 19
E6E0 7E FE 28 20 05 3E 02 32
E6E8 F8 DF 2A 0C E0 3E 2A 01
E6F0 28 DF CD 69 E3 23 CD 69
E6F8 E3 3E 00 32 0E E0 69 2A
E700 0C E0 01 00 3D 3E 2D CD
E708 69 E3 21 DF 00 22 0C E0
E710 C9 3A F7 DF FE 00 C2 C8
E718 E7 3A F6 DF FE 01 28 5C
E720 2A F4 DF 2B 01 00 40 3E
E728 2D 11 02 02 CD 5A E6 01
E730 A8 DE 3E 28 23 11 02 01
E738 CD 5A E6 11 40 58 19 7E
E740 FE 2D CA C2 E7 FE 2B 28
E748 79 FE 0F 28 75 3E 00 DB
E750 FE CB 57 20 0E 06 01 CB
E758 5F 20 02 06 81 78 32 F7
E760 DF 18 09 CB 5F 20 05 3E
E768 01 32 F6 DF 2A F4 DF 11
E770 BF 00 ED 52 30 05 3E 01
E778 32 F8 DF C9 11 03 00 21
E780 0C 06 CD B5 03 2A F4 DF
E788 11 21 58 19 7E FE 2B 28
E790 31 FE 2C 28 27 FE 2A 28
E798 29 11 20 00 19 7E FE 0F
E7A0 28 20 FE 2D 28 1C FE 2B
E7A8 28 18 2A F4 DF 3E 28 01
E7B0 B8 DE 11 02 02 CD 5A E6
E7B8 23 22 F4 DF 3E 00 32 F6
E7C0 DF C9 3E 02 32 F8 DF C9
E7C8 11 06 00 21 F7 02 CD B5
E7D0 03 3A F7 DF FE 01 20 1B
E7D8 3C 32 F7 DF 2A F4 DF 11
E7E0 20 00 ED 52 22 F4 DF 01
E7E8 D8 DE 3E 28 11 03 01 CD
E7F0 56 E6 C9 FE 02 20 1F 3C
E7F8 32 F7 DF 2A F4 DF 01 A8
E800 DE 3E 28 11 02 01 CD 5A
E808 E6 11 40 00 19 3E 2D 01
E810 00 3D CD 69 E3 C9 FE 03
E818 20 1A 3C 32 F7 DF 2A F4
E820 DF 01 D8 DE 3E 28 11 03
E828 01 CD 5A E6 11 20 00 19
E830 22 F4 DF C9 FE 04 20 18
E838 3E 00 32 F7 DF 2A F4 DF
E840 11 20 00 ED 52 01 00 3D
E848 3E 2D CD 69 E3 C3 11 E7
E850 FE 81 20 33 3C 32 F7 DF
E858 2A F4 DF 11 21 58 19 7E
E860 FE 2B CA C2 E7 FE 2C 20
E868 07 2A F4 DF 2B 22 F4 DF
E870 2A F4 DF 11 20 00 ED 52
E878 22 F4 DF 01 F0 DE 11 03
E880 02 3E 28 CD 5A E6 C9 FE
E888 84 28 5B 3C 32 F7 DF 3A
E890 F6 DF FE 01 28 23 2A F4
E898 DF 01 00 40 3E 2D 11 03
E8A0 02 CD 5A E6 23 22 F4 DF
E8A8 01 A8 DE 3E 28 11 02 01
E8B0 CD 5A E6 3E 01 32 F6 DF
E8B8 C9 2A F4 DF 01 B8 DE 3E
E8C0 28 11 02 02 CD 5A E6 23
E8C8 22 F4 DF 11 40 58 19 7E
E8D0 FE 2C 20 0C 3E 00 32 F7
E8D8 DF 3E 04 06 05 CD FC E9
E8E0 3E 00 32 F6 DF C9 3E 00
E8E8 32 F7 DF 2A F4 DF 2B 01
E8F0 00 40 3E 2D 11 02 02 CD
E8F8 5A E6 11 21 00 19 22 F4
E900 DF C3 11 E7 11 C4 00 21
[illegible] 3E 04 CD B5 03 11 83 00
E910 21 6E 06 CD B5 03 2A F4
E918 DF 01 11 3D 3E 2D CD 69
E920 E3 11 20 00 19 22 F4 DF
E928 01 A8 DE 3E 28 11 02 01
E930 CD 5A E6 11 1E 00 21 6E
E938 03 ED 5F 6F CD B5 03 2A
E940 F4 DF 11 C0 02 ED 52 38
E948 CD 3E 09 32 65 EA 3A FF
E950 DF 3D 32 FF DF C2 EE E4
E958 21 4A 01 3E 8E 06 0B DD
E960 21 88 E2 CD 2B E3 CD 3B
E968 E6 11 05 01 21 6E 06 CD
E970 B5 03 11 88 01 21 3E 04
E978 CD B5 03 11 05 01 21 6E
E980 06 CD B5 03 3E 32 32 6E
E988 E5 C3 D0 E4 11 0B 02 21
E990 26 03 CD B5 03 3A 00 E0
```

```
E998 3C CB 97 32 00 E0 3A 6E
E9A0 E5 3D 32 6E E5 3E 03 06
E9A8 05 CD FC E9 C3 E9 E4 3A
E9B0 00 E0 FE 02 30 01 C9 DD
E9B8 21 06 E0 21 A9 01 CD CE
E9C0 E9 21 71 01 CD CE E9 21
E9C8 1A 01 CD CE E9 C9 E5 ED
E9D0 5B 0A E0 ED 52 E1 38 01
E9D8 C9 DD 7E 00 3C CB A7 DD
E9E0 77 00 DD 23 FE 07 30 01
E9E8 C9 01 88 DF 16 2B FE 0F
E9F0 20 05 01 00 3D 16 2D 7A
E9F8 CD 69 E3 C9 DD 21 F9 DF
EA00 16 00 5F DD 19 DD E5 CD
EA08 12 EA DD EA 10 F7 CD 3B
EA10 E6 C9 DD E7 00 3C FE 0A
EA18 20 09 3E 00 DD 77 00 DD
EA20 2B 18 EF DD 77 00 C9 00
EA28 00 00 00 00 00 00 00 00
EA30 00 00 00 00 00 00 00 00
EA38 00 00 00 00 00 00 00 00
EA40 00 00 00 00 00 00 00 00
EA48 00 00 00 00 00 00 00 00
EA50 00 00 00 00 00 00 00 00
EA58 00 00 00 00 00 00 00 00
EA60 DD 21 0F E0 06 13 C5 DD
EA68 56 01 DD 5E 00 DD 66 03
EA70 DD 6E 02 DD E5 CD B5 03
EA78 DD E1 11 04 00 DD 19 C1
EA80 10 E4 C9 00 00 00 00 00
```



Você já deve ter observado que a rotina da música funciona independentemente de qualquer outra e foi montada distante do programa principal. Isso não interfere com o Assembler de *INPUT*, mas, se você usou outro montador, poderá descobrir que a rotina da música está na região de memória ocupada pelo próprio Assembler. Nesse caso, coloque uma nova origem na rotina da música. Tente usar 24000 em vez de 30000 — e não se esqueça de mudar as declarações **EQU** e as instruções **JSR 30000**, para que o processador vá para o novo endereço inicial quando a rotina da música for chamada.

Use o monitor de código de máquina do seu computador para ler (com **PEEK**) a memória e verificar se ela confere com a listagem hexa dada a seguir.

```
4268 03 0C 09 06 06 08 01 0E
4270 07 05 12 03 12 05 01 14
4278 05 04 20 02 19 20 01 2E
4280 04 0F 05 17 12 09 14 14
4288 05 0E 20 02 19 20 13 2E
4290 0B 05 0C 0C 01 17 01 19
4298 01 0E 04 20 07 2E 08 05
42A0 04 0C 05 19 20 01 06 14
42A8 05 12 20 01 20 13 08 0F
42B0 12 14 20 17 01 0C 0B 20
42B8 17 09 0C 0C 09 05 20 20
42C0 20 20 20 20 12 05 14 05
42C8 12 0E 13 20 14 0F 20 06
42D0 09 0E 04 20 14 08 05 20
42D8 07 0F 01 14 13 20 08 01
42E0 16 05 20 20 13 10 12 05
42E8 01 04 20 08 09 13 20 10
42F0 09 03 0F 09 03 20 07 0F
42F8 0F 04 09 05 13 20 01 0C
4300 0C 20 20 20 0F 16 05 12
4308 20 01 20 12 0F 03 0B 19
4310 20 05 0D 02 01 0E 0B 0D
4318 05 0E 14 2E 20 20 20 20
4320 20 20 20 20 20 17 09 0C
4328 0C 09 05 20 13 05 14 13
4330 20 0F 06 06 20 14 0F 20
4338 12 05 03 0C 01 09 0D 20
4340 08 09 13 20 0C 0F 13 14
4348 20 10 0F 13 13 05 13 13
4350 09 0F 0E 13 2C 02 15 14
4358 20 09 13 20 08 01 0D 10
4360 05 12 05 04 02 19 20 06
4368 01 0C 0C 09 0E 07 20 02
4370 0F 15 0C 04 05 12 13 2C
4378 10 0F 14 20 08 0F 0C 05
4380 13 2C 20 20 01 0E 04 20
4388 16 09 03 09 0F 15 13 20
4390 13 0E 01 0B 05 13 2E 14
4398 0F 20 0D 01 0B 05 20 20
43A0 20 20 20 20 0D 01 14 14
43A8 05 12 13 20 17 0F 12 13
43B0 05 20 14 08 05 20 14 09
43B8 04 05 20 09 13 20 12 09
43C0 13 09 0E 07 01 0E 04 20
43C8 08 05 20 09 13 20 09 0E
43D0 20 04 01 0E 07 05 12 20
43D8 0F 06 20 02 05 09 0E 07
43E0 20 03 15 14 0F 06 06 2E
43E8 14 0F 20 08 05 0C 10 20
43F0 17 09 0C 0C 09 05 20 09
43F8 0E 20 08 09 13 20 11 15
4400 05 13 14 20 12 05 01 04
4408 20 14 08 05 20 06 0F 0C
4410 0C 0F 17 09 0E 07 20 01
4418 0E 04 20 10 12 05 13 13
4420 20 27 13 27 14 0F 20 13
4428 14 01 12 14 2E 0E 20 20
4430 20 20 2D 20 12 15 0E 0D
4438 20 20 20 20 2D 20 16 05
4440 12 14 09 03 01 0C 20 0A
4448 15 0D 10 02 0F 14 08 20
4450 2D 20 04 09 01 07 0F 0E
4458 01 0C 20 0A 15 0D 10 23
4460 23 23 21 23 23 21 23 23
4468 23 21 23 23 21 23 23 23
4470 23 21 23 23 21 23 23 21
4478 23 23 23 21 23 23 21 21
4480 55 54 50 50 40 40 00 00
4488 7F 5F 57 D7 F5 FF D5 75
4490 DD 77 5D D5 7D 75 5D 77
4498 F5 FD 57 75 DD 77 5D D5
44A0 FD 5F 57 D7 5D FF D5 7F
44A8 75 77 FD F7 D5 5D 75 77
44B0 55 D5 D5 5D 5D D7 F5 7D
44B8 D5 5D 5D D7 55 57 FF 7F
44C0 57 57 FD FD 55 75 D5 55
44C8 55 75 D5 55 75 75 D5 D5
44D0 5D 5D D5 D5 5D 5D D7 55
44D8 57 5D D7 5D D5 D5 5D 5D
44E0 75 D7 DD 75 7D 5D 75 D5
44E8 5D 75 5D D7 57 75 5D 5D
44F0 F5 D5 57 75 5D D5 57 75
44F8 57 55 55 D7 F7 55 55 DD
4500 57 55 55 D5 77 55 55 DF
4508 D5 D5 57 55 5D D5 57 7F
4510 F5 75 5D 55 57 75 5D F5
4518 57 5D 75 5F 5D 57 D7 55
4520 75 75 57 55 55 77 75 D5
4528 55 D7 75 D5 55 D7 75 75
4530 57 57 5D 75 57 57 5D 55
4538 55 57 5D 55 7D D7 D7 D7
4540 7D 5D 7D 5D 5D FF 7D D7
4548 5D 75 FF FD 57 FD 57 FD
4550 5D 7D DD FF 5D FF D5 7D
4558 57 FD 7F D5 FD D7 7D FF
4560 57 5D 75 75 7D D7 7D D7
4568 7D 7F D7 7F 57 57 57 5F
4570 5F 57 5F 5F 5F 5F D5 F5
4578 F5 D5 F5 F5 F5 F5 5F 5F
4580 57 57 57 57 57 57 F5 F5
4588 D5 D5 D5 D5 D5 FD 55 55
4590 55 55 55 55 55 55 57 5F
4598 5F 57 55 57 57 57 D5 F5
45A0 F5 D5 55 55 55 FD 55 55
45A8 55 55 55 55 55 55 55 55
45B0 55 55 55 55 55 55 FD 55
45B8 57 5D 75 D5 75 55 55 55
45C0 D5 75 5D 5D 5F 55 55 55
45C8 55 55 55 55 55 55 55 55
45D0 55 55 57 5F 5F 57 55 55
45D8 55 55 D5 F5 F5 D5 55 57
45E0 57 57 FD 55 57 5D 55 55
45E8 55 FD 55 55 D5 75 75 D5
45F0 75 55 55 55 55 55 5D 5D
45F8 5F 55 55 55 55 55 55 55
4600 55 55 55 55 55 55 55 55
4608 55 55 57 5F 5F 57 55 55
4610 55 55 D5 F5 F5 D5 55 55
4618 55 55 55 55 55 55 55 55
4620 55 55 55 55 55 55 55 57
4628 57 57 FD 55 57 5D 55 55
4630 55 FD 55 55 D5 75 55 55
4638 55 55 55 55 55 55 55 55
4640 55 55 55 55 55 55 75 D5
4648 75 55 55 55 55 55 5D 5D
4650 5F 55 55 55 55 55 55 55
4658 55 55 55 55 55 55 57 5F
4660 7F FF FF FF 7F 5F F5 FD
4668 FF FF FF FD F5 D5 55 55
4670 55 55 55 55 55 55 5F 7F
4678 FF FF FF 7F 5F 57 D5 F5
4680 FD FF FF FF FD F5 55 55
4688 55 55 55 55 55 55 5D 57
4690 55 55 55 55 55 55 5D 75
4698 D5 D5 D5 D5 D5 D5 57 5F
46A0 7D 7F 7F 5F 57 57 D5 F5
46A8 FD FD FD F5 D5 D5 57 57
46B0 55 55 55 55 55 55 D5 D5
46B8 F5 F5 7D 7D 5F 5F 55 55
46C0 55 55 57 57 5F 5F 7D 7D
46C8 F5 F5 D5 D5 55 55 7D 7D
46D0 F5 F5 7D 7D 5F 5F 55 55
46D8 7D 5F 7D F5 D5 D5 55 55
46E0 AA AA 56 56 AA AA 55 55
46E8 AA AA 95 95 AA AA 55 55
46F0 7F FF FF FF 7F 5F 81 15
46F8 7D FF FF FF FD F5 57 57
4700 57 5F 5F 5F 5F 5F 55 55
4708 55 D5 D5 D5 D5 D5 57 5F
4710 7F 5F 5F 5F 57 57 55 D5
4718 F5 D5 D5 D5 55 55 AA AA
4720 AA A6 99 6A AA AA AA AA
4728 AA AA A9 66 9A AA AA AA
4730 AA 6A 9A A6 A9 AA AA AA
4738 AA A6 99 6A AA AA 00 05
4740 00 00 00 00 00 00 06 1D
4A38 BD 4A CC 8E 04 21 10 8E
4A40 42 68 C6 05 BD 4A E6 8E
4A48 04 67 C6 06 BD 4A E6 8E
4A50 05 61 C6 10 BD 4A E6 8E
4A58 05 A0 C6 15 BD 4A E6 8E
```

```
4A60 05 C7 C6 0C BD 4A E6 86
4A68 05 8E FF FF 30 1F 26 FC
4A70 4A 26 F6 BD 4A CC 8E 04
4A78 00 10 8E 42 A4 5F BD 4A
4A80 E6 C6 89 BD 4A E6 30 88
4A88 18 C6 0A BD 4A E6 30 88
4A90 16 C6 14 BD 4A E6 30 0C
4A98 C6 14 BD 4A E6 BD 80 06
4AA0 81 53 26 F9 39 BD 4A EE
4AA8 BD 4A D9 8E 15 FF 10 8E
4AB0 44 5F C6 20 34 04 BD 4A
4AB8 FD BD 4B 12 35 04 5A 26
4AC0 F3 10 8E 44 C4 8E 06 21
4AC8 BD 4B 43 39 8E 04 00 86
4AD0 80 A7 80 8C 06 00 25 F9
4AD8 39 8E 06 00 86 55 A7 80
4AE3 8C 1E 00 25 F9 39 A6 A0
4AE8 A7 80 5A 26 F9 39 86 E5
4AF0 B7 FF 22 B7 FF C3 B7 FF
4AF8 C5 B7 FF C7 39 34 30 8E
4B00 06 00 10 8E 06 01 A6 A0
4B08 A7 80 8C 1D FF 25 F7 35
4B10 30 39 34 10 A6 A0 80 2.
4B18 26 1C 35 10 30 89 FF 00
4B20 34 10 34 20 10 8E 44 80
4B28 C6 08 A6 A0 A7 84 30 88
4B30 20 5A 26 F6 35 20 6F 84
4B38 30 88 20 8C 1E 00 25 F6
4B40 35 10 39 C6 1E 34 04 C6
4B48 04 A6 A0 A7 80 5A 26 F9
4B50 30 88 1C 35 04 5A 26 ED
4B58 39 B6 47 3E C6 10 3D C3
4B60 46 DE 1F 03 8E 0A DE BD
4B68 4B CA B6 47 3F 27 0E 34
4B70 02 BD 4B 7E 35 02 8. 01
4B78 27 03 BD 4B 9A 39 8E 13
4B80 E7 CE 0B FF BD 4B B6 8E
4B88 11 EF CE 0B FF BD 4B B6
4B90 8E 0E F9 CE 0B FF BD 48
4B98 B6 39 8E 13 E7 CE 46 9E
4BA0 BD 4B B6 8E 11 EF CE 46
4BA8 9E BD 4B B6 8E 0E F9 CE
4BB0 46 9E BD 4B B6 39 C6 04
4BB8 34 54 BD 4B CA 35 54 30
4BC0 89 01 00 33 CB 10 5A 26
4BC8 EF 39 C6 02 34 14 C6 08
4BD0 37 02 A7 84 30 88 20 5A
4BD8 26 F6 35 14 30 01 5A 26
4BE0 EB 39 BD 4A 38 86 05 B7
4BE8 47 3F 7F 47 3E 8E 47 40
4BF0 C6 06 6F 80 5A 26 FB 86
4BF8 06 B7 47 46 8E 1D 00 BF
4C00 47 47 8E 13 E0 BF 47 49
4C08 7F 47 4B 7F 47 4C 8E 0B
4C10 FE BF 47 4D 7F 47 4F 86
4C18 05 B7 47 50 86 0A B7 47
4C20 51 BD 4A A5 8E 07 0F CE
4C28 44 88 C6 05 86 03 37 20
4C30 10 AF 81 4A 26 F8 30 88
4C38 1A 5A 26 F0 BD 4C 77 8E
4C40 08 0F CE 44 A6 C6 05 86
4C48 03 37 20 10 AF 81 4A 26
4C50 F8 30 88 1A 5A 26 F0 B6
4C58 47 3F C6 05 3D C3 45 3C
4C60 1F 03 8E 08 16 C6 05 37
4C68 02 A7 84 30 88 20 5A 26
4C70 F6 BD 75 30 7E 51 C4 34
4C78 36 8E 47 40 C6 06 10 8E
4C80 07 16 A6 84 34 14 C5 01
4C88 26 24 C6 05 3D C3 45 3C
4C90 1F 01 34 20 C6 05 A6 80
4C98 A7 A4 31 A8 20 5A 26 F6
4CA0 35 20 31 21 35 14 30 01
4CA8 5A 26 D7 35 36 39 C6 05
4CB0 3D C3 45 3C 1F 01 34 20
4CB8 C6 05 A6 80 34 02 84 0F
4CC0 48 48 48 48 8A 05 A7 21
4CC8 35 02 44 44 44 44 8A 50
4CD0 A7 A4 31 A8 20 5A 26 E2
4CD8 35 20 31 22 20 C6 CE 47
4CE0 1E B6 47 46 85 02 27 03
4CE8 CE 47 2E BE 47 47 86 10
4CF0 34 42 BD 4B CA 35 42 4A
4CF8 26 F6 7A 47 46 26 0F 86
4D00 0A B7 47 46 BE 47 47 30
4D08 89 FF 00 BF 47 47 39 7A
4D10 47 52 26 30 86 05 27 47
4D18 52 13 8E 06 21 86 1E 34
4D20 02 86 02 1C FE 34 01 5F
4D28 35 01 66 85 34 01 5C C1
4D30 8E 26 F5 68 84 35 01 66
4D38 84 4A 26 E7 30 88 20 35
4D40 02 4A 26 DB 39 B6 47 3F
4D48 27 05 81 03 27 01 39 BE
4D50 47 4D CE 06 00 34 10 BD
4D58 4B CA 35 10 30 1F 8C 14
4D60 E0 27 4B BF 47 4D A6 84
4D68 81 AA 27 42 81 55 27 09
4D70 81 5D 27 05 86 02 27 47
4D78 4C 30 89 01 21 A6 84 81
4D80 AA 27 2B 81 55 26 06 30
4D88 88 DF BF 47 4D B6 47 54
4D90 27 02 BE 47 4D CE 46 76
4D98 BD 4B CA 7F 47 54 39 BE
4DA0 47 4D CE 46 5E BD 4B CA
4DA8 86 01 B7 47 54 39 BE 47
4DB0 4D CE 06 00 BD 4B CA 8E
4DB8 0B FE BF 47 4D 39 FC 47
4DC0 49 C4 1F C1 1E 26 05 86
4DC8 01 B7 47 4C B6 47 55 10
4DD0 26 00 D9 BE 47 49 30 89
4DD8 02 20 AE 84 8C 55 55 10
4DE0 27 00 BB 8C AA AA 10 27
4DE8 00 B4 8C 5F F5 10 27 00
4DF0 AD 5F 7F 47 58 86 BF B7
4DF8 FF 02 B6 FF 00 B7 47 56
4E00 86 DF B7 FF 02 B6 FF 00
4E08 B7 47 57 81 FD 26 1A C6
4E10 01 B6 47 56 81 FD 26 02
4E18 C6 81 F7 47 55 B6 47 58
4E20 26 15 BE 47 49 34 10 20
4E28 3C B6 47 56 81 FD 26 EA
4E30 86 01 B7 47 58 20 E3 BE
4E38 47 49 CE 06 00 BD 4B CA
4E40 30 89 00 FE BD 4B CA BE
4E48 47 49 30 01 34 10 30 89
4E50 01 61 A6 84 81 D5 D7 44
4E58 81 FF 27 40 81 50 27 44
4E60 86 47 48 27 .6 AE E4 CE
4E68 45 CE BD 4B CA AE E4 30
4E70 89 01 00 BD 4B CA 7F 47
4E78 4B 20 19 AE E4 CE 45 96
4E80 BD 4B CA AE E4 30 89 01
4E88 00 CE 45 B6 BD 4B CA 86
4E90 01 B7 47 4B AE E4 BF 47
4E98 49 35 10 39 35 10 86 02
4EA0 B7 47 4C 39 35 10 30 1F
4EA8 34 10 20 B9 BD 51 6F B6
4EB0 47 55 81 01 26 2C 7C 47
4EB8 55 BE 47 49 34 10 30 89
4EC0 01 00 CE 06 00 BD 4B CA
4EC8 35 10 30 89 FF 00 BF 47
4ED0 49 CE 45 96 BD 4B CA 30
4ED8 89 00 FE CE 45 B6 BD 4B
4EE0 CA 39 81 02 26 1F 7C 47
4EE8 55 BE 47 49 30 89 FF 00
4EF0 CE 45 CE BD 4B CA 30 89
4EF8 00 FE BD 4B CA 30 89 00
4F00 FE BD 4B CA 39 81 03 26
4F08 2F 7C 47 55 BE 47 49 30
4F10 89 FF 00 CE 06 00 BD 4B
4F18 CA 30 89 00 FE CE 46 06
4F20 BD 4B CA 30 89 00 FE CE
4F28 46 26 BD 4B CA 30 89 00
4F30 FE CE 46 46 BD 4B CA 39
4F38 81 04 26 25 7F 47 55 BE
4F40 47 49 34 10 CE 06 00 BD
4F48 4B CA 35 10 30 89 01 00
4F50 BF 47 49 CE 45 6E BD 4B
4F58 CA 30 89 00 FE BD 4B CA
4F60 39 81 81 26 30 7C 47 55
4F68 BE 47 49 30 89 01 22 A6
4F70 84 81 57 10 27 FF 27 BD
4F78 50 3F BE 47 49 30 89 FF
4F80 01 BF 47 49 CE 45 96 BD
4F88 4B CA 30 89 00 FE CE 45
4F90 B6 BD 4B CA 39 81 82 26
4F98 40 7C 47 55 BD 50 3F BE
4FA0 47 49 30 89 FF 01 BF 47
4FA8 49 CE 45 CE BD 4B CA 30
4FB0 89 00 FE BD 4B CA 30 89
4FB8 00 FE BD 4B CA BE 47 49
4FC0 30 89 03 60 A6 80 81 FF
4FC8 27 07 A6 84 81 FF 27 01
4FD0 39 86 04 C6 05 BD 51 0F
4FD8 39 81 83 26 2B 7C 47 55
4FE0 BD 50 3F 30 89 00 FE CE
4FE8 06 00 BD 4B CA BE 47 49
4FF0 30 89 01 01 BF 47 49 CE
4FF8 45 96 BD 4B CA 30 89 00
5000 FE CE 45 B6 BD 4B CA 39
5008 81 84 26 1C BD 50 3F BE
5010 47 49 34 10 30 89 02 00
```



```
5018 10 AE 84 35 10 10 8C 55
5020 55 26 05 30 01 BF 47 49
5028 7F 47 55 7F 47 4B BE 47
5030 49 CE 45 6E BD 4B CA 30
5038 89 00 FE BD 4B CA 39 BE
5040 47 49 CE 06 00 BD 4B CA
5048 30 89 00 FE BD 4B CA 39
5050 86 88 8E 00 8C BD 51 33
5058 86 83 8E 00 D5 BD 51 33
5060 BE 47 49 CE 06 00 BD 4B
5068 CA 30 89 00 FE BF 47 49
5070 CE 45 6E BD 4B CA 30 89
5078 00 FE BD 4B CA 86 1E 8E
5080 00 71 BD 51 33 DE 47 49
5088 8C 1B 00 25 D3 7A 47 3F
5090 10 26 FB 63 86 05 8E FF
5098 FF 30 1F 26 FC 4A 26 F6
50A0 BD 4A CC B6 FF 22 84 0F
50A8 B7 FF 22 B7 FF C2 B7 FF
50B0 C4 B7 FF C6 10 8E 05 0B
50B8 8E 07 01 AF A1 8E 0D 05
50C0 AF A1 8E 20 0F AF A1 8E
50C8 16 05 AF A1 8E 12 21 AF
50D0 A1 86 C8 8E 00 FF BD 51
50D8 33 86 C8 8E 00 C8 BD 51
50E0 33 86 FF 8E 00 FF BD 51
50E8 33 86 64 D7 51 EE 16 FA
50F0 F1 86 FF 8E 00 96 BD 51
50F8 33 B6 47 3E 4C 84 03 B7
5100 47 3E 7A 51 EE C6 05 86
5108 03 BD 51 0F 16 FA E8 1E
5110 89 8E 47 40 3A 34 12 BD
5118 51 23 35 12 4A 26 F2 BD
5120 4C 77 39 A6 84 4C 81 0A
5128 26 06 6F 84 30 1F 20 F3
5130 AF 84 39 34 02 B6 FF 01
5138 84 FF B7 FF 01 B6 FF 03
```

```
5140 84 F7 B7 FF 03 B6 FF 23
5148 8A 08 B7 FF 23 1A 50 35
5150 02 34 10 C6 FC F7 FF 20
5158 30 1F 26 FC AE E4 7F FF
5160 20 30 1F 26 FC AE E4 4A
5168 26 EB 1C AF 35 10 39 8E
5170 00 62 86 04 BD 51 33 39
5178 B6 47 E3 81 02 24 01 39
5180 10 8E 47 4F 8E 13 E7 BD
5188 51 97 8E 11 EF BD 51 97
5190 8E 0E F9 BD 51 97 39 34
5198 10 BE 47 47 30 88 1F AC
51A0 E4 35 10 22 01 39 A6 A4
51A8 4C 84 0F A7 A0 81 07 24
51B0 01 39 CE 46 8E 81 0F 26
51B8 03 CE 06 00 30 89 FF 00
51C0 BD 4B CA 39 86 05 B7 47
51C8 52 7F 47 55 BD 4B 59 BD
51D0 4D BE BD 4D 45 BD 51 78
51D8 BD 4C DE BD 4D 0F B6 47
51E0 4C 81 01 10 27 FF 0A 81
51E8 02 10 27 FE 63 C6 64 4F
51F0 4A 26 CD 5A 26 F9 BD 80
51F8 06 81 03 26 D2 39 39 03
5200 00 00 00 00 00 00 00 00
7530 8E 75 8A BF 75 88 86 13
7538 34 02 B6 FF 01 84 F7 B7
7540 FF 01 B6 FF 03 84 F7 B7
7548 FF 03 B6 FF 23 8A 08 B7
7550 FF 23 FE 75 88 1A 50 37
7558 12 11 83 75 CD 25 03 CE
7560 75 8A FF 75 88 34 10 C6
7568 FC F7 FF 20 30 1F 26 FC
7570 AE E4 7F FF 20 30 1F 26
7578 FC AE E4 4A 26 EB 32 62
7580 6A E4 26 CE 1C AF 35 82
7588 75 C3 62 00 BD E9 00 9E
```

```
7590 83 00 8D DC 00 7D 4E 00
7598 76 93 00 7D FF 00 8D 6E
75A0 00 A8 83 00 D4 31 00 BD
75A8 6E 00 A8 E9 00 9E 62 00
75B0 BD 93 00 BD 2C 00 D4 62
75B8 00 BD DC 00 A8 5C 00 C8
75C0 DC 00 FC 00 00 00 00 00
75C8 00 00 00 00 00 00 00 00
```



Avalanche foi montado a partir do endereço 49436, para possibilitar a utilização do Assembler de *INPUT*.

Provavelmente você já percebeu que alguns trechos da memória do computador não foram aproveitados: entre a tabela de cores e a tabela de padrões e entre o fim do programa e a tabela de perfil. Assim, você terá a possibilidade de alterar os endereços de montagem tanto da tabela de cores como da tabela de perfil, mas não se esqueça de alterar também todas as instruções nas rotinas que fazem referência a essas tabelas. Essa precaução deve ser tomada sempre que quiser modificar determinado endereço de montagem no programa.

Faça um programa BASIC ou examine manualmente as posições da memória, verificando se os conteúdos conferem com a listagem hexadecimal a seguir. Se você tiver deslocado alguma parte do jogo, os endereços serão diferentes, mas a seqüência de códigos hexa deve ser exatamente a mesma.

```
C11C 17 17 17 17 17 17 17 17
C124 17 17 17 17 17 17 17 17
C12C 17 17 17 17 17 17 17 17
C134 17 17 17 17 17 17 17 17
C13C 17 17 17 17 17 17 17 17
C144 17 17 17 17 17 17 17 17
C14C 17 17 17 17 17 17 17 17
C154 17 17 17 17 17 17 17 17
C15C 17 17 17 17 17 17 17 17
C164 17 17 17 17 17 17 17 17
C16C 17 17 17 17 17 17 17 17
C174 17 17 17 17 17 17 17 17
C17C 67 67 67 67 67 67 67 67
C184 67 67 67 67 67 67 67 67
C18C 67 67 67 67 67 67 67 67
C194 67 67 67 67 67 67 67 67
C19C 67 67 67 67 67 67 67 67
C1A4 67 67 67 67 67 67 67 67
C1AC 67 67 67 67 67 67 67 67
C1B4 67 67 67 67 67 67 67 67
C1BC F7 F7 F7 F7 F7 F7 F7 F7
C1C4 F7 F7 F7 F7 F7 F7 F7 F7
C1CC F7 F7 F7 F7 F7 F7 F7 F7
C1D4 F7 F7 F7 F7 F7 F7 F7 F7
C1DC F7 F7 F7 F7 F7 F7 F7 F7
C1E4 F7 F7 F7 F7 F7 F7 F7 F7
C1EC F7 F7 F7 F7 F7 F7 F7 F7
C1F4 F7 F7 F7 F7 F7 F7 F7 F7
C1FC 17 17 17 17 17 17 17 17
C204 17 17 17 17 17 17 17 17
C20C 17 17 17 17 17 17 17 17
C214 17 17 17 17 17 17 17 17
```

C21C	17	17	17	17	17	17	17	17
C224	17	17	17	17	17	17	17	17
C22C	17	17	17	17	17	17	17	17
C234	17	17	17	17	17	17	17	17
C23C	C7	C7	C7	C7	C7	C7	C7	[illegible]
C244	C7	C7	C7	C7	C7	C7	C7	C7
C24C	C7	C7	C7	C7	C7	C7	C7	C7
C254	C7	[illegible]	C7	C7	C7	C7	C7	C7
C25C	C7	C7	C7	C7	C7	C7	C7	C7
C264	C7	C7	C7	C7	C7	C7	C7	C7
C26C	C7	C7	C7	C7	C7	C7	C7	C7
C274	C7	C7	C7	C7	C7	C7	C7	C7
C27C	C7	C7	C7	C7	C7	C7	C7	C7
C284	C7	C7	C7	C7	C7	C7	C7	C7
C28C	C7	C7	C7	C7	C7	C7	C7	C7
C294	C7	C7	C7	C7	C7	C7	C7	C7
C29C	37	37	37	37	37	37	37	37
C2A4	37	37	37	37	37	37	37	37
C2AC	37	37	37	37	37	37	37	37
C2B4	37	37	37	37	37	37	37	37
C2BC	37	37	37	37	37	37	37	37
C2C4	37	37	37	37	37	37	37	37
C2CC	37	37	37	37	37	37	37	37
C2D4	37	37	37	37	37	37	37	37
C2DC	A7	A7	A7	A7	A7	A7	A7	A7
C2E4	A7	A7	A7	A7	A7	A7	A7	A7
C2EC	A7	A7	A7	A7	A7	A7	A7	A7
C2F4	A7	A7	A7	A7	A7	A7	A7	A7
C2FC	87	87	87	87	87	87	87	87
C304	87	87	87	87	87	87	87	87
C30C	87	87	87	87	87	87	87	87
C314	87	87	87	87	87	87	87	87
C31C	D7	D7	D7	D7	D7	D7	D7	D7
C324	D7	D7	D7	D7	D7	D7	D7	D7
C32C	D7	D7	D7	D7	D7	D7	D7	D7
C334	D7	D7	D7	D7	D7	D7	D7	D7
C33C	97	97	97	97	97	97	97	97
C344	97	[illegible]	97	97	97	97	97	97
C34C	97	97	97	97	97	97	97	97
C354	97	[illegible]	97	97	97	97	97	97
C35C	F4	F4	F4	F4	F4	F4	F4	F4
C364	F4	F4	F4	F4	F4	F4	F4	F4
C36C	F4	F4	F4	F4	F4	F4	F4	F4
C374	F4	F4	F4	F4	F4	F4	F4	F4
C37C	F4	F4	F4	F4	F4	F4	F4	F4
C384	F4	F4	F4	F4	F4	F4	F4	F4
C38C	F4	F4	F4	F4	F4	F4	F4	F4
C394	F4	F4	F4	F4	F4	F4	F4	F4
C39C	33	33	33	33	33	33	33	33
C3A4	33	33	33	33	33	33	33	33
C3AC	33	33	33	33	33	33	33	33
C3B4	33	33	33	33	33	33	33	33
C3BC	[illegible]	08	43	4C	41	20	00	00
C3C4	00	00	00	00	08	58	33	43
C3CC	[illegible]	30	00	00	00	00	00	08
C3D4	[illegible]	33	43	15	70	00	00	00
C3DC	[illegible]	00	03	42	00	01	28	F1
C3E4	[illegible]	00	FF	FF	00	00	FF	FF
C3EC	[illegible]	00	FE	FF	00	00	FF	FF
C3F4	00	00	FF	FF	00	00	FF	FF
C3FC	[illegible]	00	FF	FF	FE	FF	00	00
C404	FF	FF	00	00	FE	FF	00	00
C40C	FF	FF	C0	00	FF	FF	00	00
C414	FF	FF	00	00	FF	FF	00	00
C41C	FF	FF	00	00	FF	FF	00	00
C424	FF	FF	00	00	FF	FF	24	12
C42C	24	12	AE	0C	24	12	24	12
C434	AE	0C	02	0D	48	6C	[illegible]	00
C43C	54	54	0F	24	12	AF	0C	E8
C444	0C	48	[illegible]	D3	00	54	F0	F3
C44C	00	0C	00	48	6C	C4	20	11
C454	00	2E	FC	71	03	4B	07	18
C45C	01	CF	46	01	00	C8	7D	F2
C464	2E	BB	46	49	C5	2D	94	F3

C46C	6B	42	00	24	6B	2D	94	DB
C474	62	F3	62	01	46	00	00	00
C47C	FF	FF	00	00	FF	FF	00	00
C484	FF	FF	C0	00	FF	FF	00	00
C48C	FF	FF	00	00	FF	FF	00	00
C494	FF	FF	00	00	FF	FF	00	00
C49C	FF	FF	00	00	18	3C	3C	18
C4A4	3C	3C	3C	3C	3C	3C	18	18
C4AC	18	18	18	1E	00	00	00	00
C4B4	00	00	00	00	00	00	00	00
C4BC	00	00	00	00	01	03	03	01
C4C4	00	01	01	01	0F	00	01	02
C4CC	04	08	04	00	80	C0	C0	80
C4D4	00	00	00	E0	00	00	80	40
C4DC	20	20	30	00	00	00	00	00
C4E4	00	00	00	00	00	00	00	00
C4EC	00	00	00	00	18	3C	3C	18
C4F4	00	10	10	1E	E0	00	0C	24
C4FC	42	82	43	00	00	00	00	00
C504	00	00	00	00	1C	3E	7F	FF
C50C	FF	FE	FC	38	00	00	00	00
C514	00	00	00	00	00	00	00	00
C51C	00	00	00	00	00	00	00	00
C524	00	00	00	00	03	07	0F	0F
C52C	0F	07	03	01	00	00	00	00
C534	00	00	00	00	80	C0	E0	F0
C53C	F0	F0	E0	C0	00	00	07	18
C544	20	40	40	80	80	40	7C	02
C54C	02	01	00	00	00	00	1F	A0
C554	C0	00	00	00	00	00	00	04
C55C	0A	11	E0	00	00	00	80	40
C564	5C	22	02	02	02	04	08	04
C56C	04	04	F8	00	00	00	00	00
C574	00	00	00	00	00	00	00	00
C57C	00	00	00	00	00	00	78	86
C584	01	01	00	00	00	00	00	00
C58C	00	00	00	00	00	00	1E	61
C594	80	80	00	00	00	00	00	00
C59C	00	00	00	00	00	00	00	00
C5A4	87	79	00	00	00	00	00	00
C5AC	00	00	00	00	00	00	00	00
C5B4	[illegible]	9F	00	00	00	00	00	00
C5BC	00	00	00	00	22	14	08	08
C5C4	08	08	08	08	18	3C	36	7E
C5CC	7F	3C	18	18	00	00	00	00
C5D4	00	00	00	00	00	00	00	00
C5DC	00	00	00	00	18	18	0C	0C
C5E4	06	06	03	03	06	06	0C	0C
C5EC	18	18	30	30	00	00	00	00
C5F4	00	00	00	00	00	00	00	00
C5FC	00	00	00	00	60	60	C6	C3
C604	66	6C	38	18	00	00	00	00
C60C	00	00	00	00	00	00	00	00
C614	00	00	00	00	00	00	00	00
C61C	00	00	00	00	84	D6	FF	FF
C624	FF	FF	FF	FF	00	00	00	00
C62C	00	00	00	00	00	00	00	00
C634	00	00	00	00	00	00	00	00
C63C	00	00	00	00	00	01	03	07
C644	1F	3F	7F	FF	00	00	00	00
C64C	00	00	00	00	00	00	00	00
C654	00	00	00	00	00	00	00	00
C65C	00	00	00	00	00	00	FF	FF
C664	3C	3C	FF	FF	00	00	00	00
C66C	00	00	00	00	00	00	00	00
C674	00	00	00	00	00	00	00	00
C67C	00	00	00	00	06	08	76	FF
C684	FF	FF	7E	3C	00	00	00	00
C68C	00	00	00	00	00	00	00	00
C694	00	00	00	00	00	00	00	00
C69C	00	00	00	00	10	10	10	38
C6A4	38	38	38	38	00	00	00	00
C6AC	00	00	00	00	00	00	00	00
C6B4	00	00	00	00	00	00	00	00

C6BC	00	00	00	00	10	38	7C	38
C6C4	38	38	10	10	00	00	00	00
C6CC	00	00	00	00	00	00	00	00
C6D4	00	00	00	00	00	00	00	00
C6DC	00	00	00	00	00	00	00	20
C6E4	51	8A	04	00	00	00	00	00
C6EC	00	00	00	00	00	00	00	00
C6F4	00	00	00	00	00	00	00	00
C6FC	00	00	00	00	00	00	00	82
C704	45	28	10	00	00	00	00	[illegible]
C70C	00	00	00	00	00	00	00	[illegible]
C714	00	00	00	00	00	00	[illegible]	00
C71C	00	00	00	00	53	43	4F	52
C724	45	20	30	30	30	30	[illegible]	30
C72C	20	20	56	49	44	41	53	20
C734	30	20	20	4E	49	[illegible]	45	[illegible]
C73C	20	30	4E	4F	56	41	20	50
C744	41	52	54	49	44	41	[illegible]	[illegible]
C74C	32	33	34	35	36	37	[illegible]	[illegible]
C754	00	00	00	A6	30	BA	30	95
C75C	07	18	08	0F	09	0F	FD	[illegible]
C764	FD	00	0A	D5	00	FD	00	1E
C76C	BD	00	D5	00	0A	A9	00	[illegible]
C774	00	0F	9F	00	D5	00	[illegible]	[illegible]
C77C	00	1C	01	0A	BD	00	1C	[illegible]
C784	1E	E1	00	E1	00	0A	[illegible]	01
C78C	E1	00	0F	FD	00	E1	00	05
C794	E1	00	FD	00	0A	D5	00	FD
C79C	00	1E	FD	00	FD	00	0A	FD
C7A4	00	3F	01	0F	0C	[illegible]	[illegible]	01
C7AC	05	FD	00	3F	01	[illegible]	E1	00
C7B4	FD	00	11	E1	00	A9	00	0A
C7BC	0C	01	E1	00	0A	A9	00	52
C7C4	01	14	FD	00	52	01	0A	05
C7CC	00	FD	00	1E	BD	00	FD	00
C7D4	0A	A9	00	D5	00	0F	9F	00
C7DC	D5	00	05	A9	00	D5	00	0A
C7E4	BD	00	1C	01	1E	E1	00	1C
C7EC	01	0A	1C	01	E1	00	0F	FD
C7F4	00	E1	00	05	E1	00	E1	00
C7FC	0A	D5	00	FD	00	1E	E1	00
C804	E1	00	05	FD	00	D5	00	05
C80C	FD	00	BD	00	05	0C	01	A9
C814	00	0F	2D	01	A9	00	05	0C
C81C	01	A9	00	0A	FD	00	FD	00
C824	1E	FD	00	7E	00	0A	FD	00
C82C	A9	00	0A	FD	00	FD	00	1E
C834	8E	00	D5	00	1E	8E	00	BD
C83C	00	05	8E	00	A9	00	05	8E
C844	00	A9	00	0F	9F	00	BD	00
C84C	05	A9	00	D5	00	0A	BD	00
C854	1C	01	1E	BD	00	FD	00	05
C85C	E1	00	E1	00	0A	1C	01	E1
C864	00	0F	FD	00	FD	00	05	E1
C86C	00	1C	01	0A	D5	00	FD	00
C874	1E	D5	00	E1	00	05	FD	00
C87C	D5	00	0A	FD	00	D5	00	0F
C884	0C	01	E1	00	05	FD	00	FD
C88C	00	0A	E1	00	52	01	11	E1
C894	00	A9	00	0A	0C	01	E1	00
C89C	0A	52	01	52	01	14	52	01
C8A4	7B	01	0A	8E	00	AA	01	1E
C8AC	8E	00	7B	01	05	8E	00	52
C8B4	01	0A	8E	00	52	01	0F	9F
C8BC	00	7B	01	05	A9	00	AA	01
C8C4	0A	BD	00	1C	01	1E	BD	00
C8CC	FD	00	05	E1	00	E1	00	0A
C8D4	8E	00	E1	00	0F	FD	00	FD
C8DC	00	05	E1	00	1C	01	0A	D5
C8E4	00	FD	00	1E	E1	00	E1	00
C8EC	05	FD	00	D5	00	05	FD	00
C8F4	BD	00	05	0C	01	A9	00	0F
C8FC	2D	01	A9	00	05	0C	01	52
C904	01	0A	FD	00	FD	00	1E	FD

```
C90C 00 7E 00 0A FD 00 A9 00
C914 0A FD 00 FD 00 1E 08 00
C91C 09 00 0A 00 64 68 6C 28
C924 2D 35 32 30 30 29 28 2D
C92C 35 32 30 30 29 2C 68 6C
C934 28 2D 35 32 30 30 29 2C
C93C 68 6C 6C 64 21 6C 64 6C
C944 64 35 35 35 35 35 32 32
C94C 35 35 32 35 41 56 41 4C
C954 41 4E 43 48 45 43 72 69
C95C 61 87 B1 6F 3A 52 2E 4E
C964 65 64 65 72 50 72 6F 67
C96C 72 61 6D 61 3A 4D 2E 48
C974 75 61 73 63 61 72 20 20
C97C 20 20 20 20 41 70 A2 73
C984 20 75 6D 20 70 65 71 75
C98C 65 6E 6F 20 70 61 73 73
C994 65 69 6F 20 20 57 63 6C
C99C 69 6F 20 [illegible] 20 20
C9A4 76 6F [illegible] 74 [illegible] 20 [illegible] 6F
C9AC [illegible]
C9B4 6E 64 65 20 [illegible] 65 74
C9BC [illegible]
C9C4 [illegible]
C9CC [illegible]
C9D4 [illegible] 6E 69 71 [illegible] 65 20
C9DC [illegible] 64 65 73 63 6F 62
C9E4 [illegible] 20 71 75 65 20 75
C9EC [illegible] 20 20 20 20 20 20
C9F4 62 61 6E 64 6F 20 64 65
C9FC [illegible] 61 62 72 69 74 6F
CA04 [illegible] 6D 6F 6E 74 65 73
CA0C [illegible] 20 65 73 70 61 6C
CA14 [illegible] 20 20 20 20
CA1C [illegible] 20 73 65 75
CA24 [illegible] 63 68 65 20
CA2C [illegible] 65 6E 63 6F 73
CA34 74 [illegible] 2E 20 20 20 20 20
CA3C 20 20 20 20 20 20 20 20
CA44 20 20 20 20 20 20 20 20
CA4C 20 20 20 20 20 20 20 20
CA54 20 20 20 20 20 20 20 20
CA5C 20 20 20 20 20 20 20 20
CA64 20 20 20 20 20 20 20 20
CA6C 20 20 20 20 41 67 6F 72
CA74 61 20 65 6C 65 20 64 65
CA7C 76 65 20 73 75 62 69 72
CA84 20 61 6F 20 74 6F 70 6F
CA8C 20 64 61 20 20 20 20 20
CA94 6D 6F 6E 74 61 6E 68 61
CA9C 20 70 61 72 61 20 72 65
CAA4 63 75 70 65 72 61 72 20
CAAC 73 75 61 73 20 63 6F 69
CAB4 73 61 73 2C 20 20 20 20
CABC 65 6E 66 72 65 6E 74 61
CAC4 6E 64 6F 20 75 6D 61 20
CACC 61 76 61 6C 61 6E 63 68
CAD4 65 2C 20 63 6F 62 72 61
CADC 73 20 20 20 20 20 20 20
CAE4 76 65 6E 65 6E 6F 73 61
CAEC 73 20 65 20 63 6F 72 72
CAF4 65 6E 64 6F 20 6F 20 72
CAFC 69 73 63 6F 20 64 65 20
CB04 63 61 69 72 20 20 20 20
CB0C 6E 75 6D 20 62 75 72 61
CB14 63 6F 2E 20 20 20 20 20
CB1C 20 20 20 20 20 20 20 20
CB24 20 20 20 20 20 20 20 20
CB2C 20 20 20 20 20 20 20 20
CB34 20 20 20 20 20 20 20 20
CB3C 20 20 20 20 20 20 20 20
CB44 20 20 20 20 20 20 20 20
CB4C 20 20 20 20 20 20 20 20
```

```
CB54 20 20 20 20 20 20 20 20
CB5C [illegible] 72 61
CB64 [illegible] 72 20
CB6C 61 20 73 69 74 75 61 87
CB74 [illegible] 20 [illegible] 61
CB7C [illegible] 20 20 20
CB84 [illegible]
CB8C [illegible]
CB94 [illegible]
CB9C [illegible]
CBA4 [illegible]
CBAC [illegible]
CBB4 [illegible]
CBBC [illegible]
CBC4 [illegible] 65 [illegible] 61
CBCC [illegible] 20 20 20 20
CBD4 [illegible] 74 72 75 87 B5
CBDC [illegible] 20 70 72 65
CBE4 73 [illegible] 69 6F 6F 65 20 61
CBEC 20 74 65 63 6C 61 20 27
CBF4 [illegible] 2E 20 20 20 20 20
CBFC 20 20 20 20 20 20 20 20
CC04 20 20 20 20 20 20 20 20
CC0C 20 20 20 20 20 20 20 20
CC14 20 20 20 20 20 20 20 20
CC1C 20 20 20 20 20 20 20 20
CC24 20 20 20 20 20 20 20 20
CC2C 20 20 20 20 20 20 20 20
CC34 20 20 20 20 20 20 20 20
CC3C 20 20 20 20 20 20 20 20
CC44 20 20 20 20 20 20 20 20
CC4C 55 73 65 3A 20 20 20 20
CC54 20 20 20 20 20 20 20 20
CC5C 20 20 20 20 20 20 20 20
CC64 20 20 20 20 20 20 20 20
CC6C 20 20 20 20 20 20 20 20
CC74 20 20 20 20 20 4E 20 20
CC7C 20 20 20 20 70 61 72 61
CC84 20 20 20 20 20 43 6F 72
CC8C 72 65 72 20 20 20 20 20
CC94 20 20 20 20 20 20 20 20
CC9C 20 20 20 20 20 4D 20 20
CCA4 20 20 20 20 70 61 72 61
CCAC 20 20 20 20 20 53 61 6C
CCB4 74 61 72 20 20 20 20 20
CCBC 20 20 20 20 20 20 20 20
CCC4 20 20 20 20 20 41 6D 62
CCCC 6F 73 20 20 70 61 72 61
CCD4 20 75 6D 20 20 53 61 6C
CCDC 74 6F 20 44 69 61 67 6F
CCE4 6E 61 6C 20 20 20 20 20
CCEC 20 20 20 20 20 20 20 20
CCF4 20 20 20 20 20 20 20 20
CCFC 20 20 20 20 20 20 20 20
CD04 20 20 20 20 20 20 20 20
CD0C 20 20 20 20 20 20 20 20
CD14 20 20 20 20 20 20 20 20
CD1C 20 20 20 20 20 20 20 20
CD24 20 20 20 20 20 20 20 20
CD2C 20 20 20 20 20 20 20 20
CD34 20 20 20 20 20 20 20 20
```

```
D000 CD 6C 00 11 28 01 21 50
D008 C9 01 09 00 CD 5C 00 11
D010 ED 01 21 59 C9 01 0F 00
D018 CD 5C 00 11 3C 02 21 68
D020 C9 01 12 00 CD 5C 00 06
D028 0A ED 6B 11 00 00 2B E5
D030 ED 52 E1 20 F9 10 F2 11
D038 00 00 21 7A C9 01 C0 03
D040 CD 5C 00 CD 9F 00 FE 53
D048 20 F9 C9 21 5C C7 06 06
D050 0E A0 ED A3 0E A1 ED A3
```

```
D058 20 F6 C9 06 04 0E A1 3E
D060 00 D3 A0 3C ED A3 20 F9
D068 C9 46 21 E8 03 11 00 00
D070 2B E5 ED 52 E1 20 F9 10
D078 F1 C9 CD 4B D0 06 14 C5
D080 CD 5B D0 E5 CD 69 D0 E1
D088 23 C1 10 F3 C1 4E D0 C9
D090 21 E9 F3 36 04 23 36 07
D098 23 36 07 [illegible] E2
D0A0 32 E0 F3 CD 69 00 C9 ED
D0A8 5B CB F3 21 A0 C4 [illegible] 80
D0B0 02 C5 E5 D5 CD 5C 00 D1
D0B8 21 00 08 19 54 5D E1 [illegible]
D0C0 C5 E5 D5 CD 5C 00 D1 21
D0C8 00 08 19 54 5D E1 C1 C5
D0D0 E5 CD 5C 00 ED 5B CB F3
D0D8 E1 C1 CD 5C 00 C9 ED 5B
D0E0 C9 F3 21 1C C1 01 A0 02
D0E8 C5 E5 D5 CD 5C 00 D1 21
D0F0 00 08 19 54 5D E1 C1 C5
D0F8 E5 D5 CD 5C 00 D1 21 00
D100 08 19 54 5D E1 C1 CD 5C
D108 00 C9 2A C7 F3 11 90 F8
D110 01 00 03 CD 59 00 C9 [illegible]
D118 8F EB 21 8E EB 06 18 C5
D120 1A 01 1F 00 ED B8 12 2B
D128 1B C1 10 F3 C9 FD 5B C7
D130 F3 21 90 E8 01 00 03 CD
D138 5C 00 C9 3E FF 2A C7 F3
D140 01 00 03 CD F6 00 [illegible] A7
D148 [illegible]
D150 [illegible]
D158 [illegible]
D160 [illegible]
D168 93 EB FE 01 20 07 3A 92
D170 EB [illegible] 32
D178 93 EB [illegible]
D180 05 3E 01 32 93 EB 3A 92
D188 [illegible]
D190 [illegible]
D198 [illegible]
D1A0 [illegible]
D1A8 [illegible]
D1B0 [illegible]
D1B8 00 19 10 F9 C9 3A 94 EB
D1C0 [illegible]
D1C8 [illegible]
D1D0 [illegible]
D1D8 [illegible]
D1E0 [illegible]
D1E8 [illegible]
D1F0 [illegible]
D1F8 3E FE E5 C5 [illegible]
D200 E1 [illegible]
D208 EB C9 3E 25 32 [illegible]
D210 03 32 96 EB C6 [illegible]
D218 EB C6 03 32 98 EB [illegible]
D220 [illegible] D5 [illegible] 91 01 [illegible]
D228 01 D5 0E 03 [illegible] C7 F3 [illegible]
D230 95 EB ED 53 97 EB D1 06
D238 04 [illegible] ED 5B 99 EB [illegible] 13
D240 ED 53 99 EB E5 C5 CD 4D
D248 00 C1 E1 [illegible] 20 [illegible] 10 [illegible]
D250 CD 20 D9 C9 00 00 00 00
D258 00 00 [illegible] 3E [illegible] 32
D260 7E D0 CD 7A D0 3E 05 [illegible]
D268 9B EB 3F 00 [illegible] 94 EB [illegible]
D270 00 00 22 [illegible] EB 22 [illegible] EB
D278 22 A1 EB 3E 14 32 7E D0
D280 CD 90 D0 3E 06 32 A3 EB
D288 21 E0 02 22 A4 EB [illegible]
D290 00 22 A6 EB 3E 03 32 A8
D298 EB 3E 00 32 A9 EB 3E [illegible]
```

```
D2A0 32 AA EB 22 C7 01 22 AB
D2A8 EB 21 00 00 22 AD EB 3E
D2B0 00 32 AF EB 21 FF 00 22
D2B8 B0 EB 3E 10 06 05 32 B2
D2C0 EB 80 32 B3 EB 80 32 B4
D2C8 EB CD 3B D1 CD BD D1 3E
D2D0 A0 32 B7 FC 3E 10 32 B9
D2D8 FC 11 20 C7 06 34 C5 D5
D2E0 1A CD 8D 00 D1 13 C1 10
D2E8 F5 11 01 00 3E 54 06 1E
D2F0 C5 D5 F5 2A C7 F3 19 CD
D2F8 4D 00 F1 3C D1 13 C1 10
D300 EF CD 07 D3 C3 2E D3 11
D308 9D EB ED 53 BA EB 11 07
D310 00 2A C7 F3 19 06 06 C5
D318 E5 ED 5B BA EB 1A 13 ED
D320 53 BA EB C6 7E CD 4D 00
D328 E1 23 C1 10 EA C9 3A 9B
D330 EB C6 7E 11 15 00 2A C7
D338 F3 19 CD 4D 00 3A 94 EB
D340 C6 7E 11 1E 00 2A C7 F3
D348 19 CD 4D 00 CD 7A D0 21
D350 1A C9 CD 4E D0 CD 4B D5
D358 CD 80 D4 CD 0C D9 CD CA
D360 D3 CD FE D3 CD 8C D3 3A
D368 AF EB FE 01 CA D2 D8 FE
D370 02 CA 26 D8 06 32 3E FF
D378 3D 20 FD 10 F9 3E 07 CD
D380 41 01 CB 67 20 CF C9 C9
D388 C9 C9 FF FF 3A AA EB 3C
D390 CB 9F 32 AA EB 06 1C FE
D398 04 38 02 06 20 C5 78 2A
D3A0 C7 F3 11 2A 00 19 E5 F5
D3A8 CD 4D 00 F1 E1 23 C6 02
D3B0 [illegible] 4D 00 C1 78 2A C7 F3
D3B8 11 44 00 19 E5 F5 CD 4D
D3C0 00 F1 E1 23 C6 02 CD 4D
D3C8 [illegible] [illegible] 06 48 3A A3 EB CB
D3D0 57 28 02 0F [illegible] 2A C7 F3
D3D8 ED 5B A4 EB 19 78 01 20
D3E0 [illegible] [illegible] 5E 00 3A A3 EB 3D
D3E8 32 A3 EB 20 10 3E 0A 32
D3F0 A3 EB 2A A4 EB 11 20 00
D3F8 ED 52 22 A4 EB C9 3A A8
D400 EB 3D 32 A8 EB FE 00 28
D408 01 C9 3E 06 32 A8 EB 3A
D410 A9 EB FE 00 28 06 21 A6
D418 FB 35 18 04 21 A6 EB 34
D420 2A CD F3 06 06 3A A6 EB
D428 4F 16 14 1E 01 CD 60 D4
D430 2A CD F3 11 04 00 19 06
D438 06 3A A6 EB C6 10 4F 16
D440 18 1E 01 CD 60 D4 3A A6
D448 FB FE 02 20 06 3E 00 32
D450 A9 EB C9 3A A6 EB FE E6
D458 20 05 3E 01 32 A9 EB C9
D460 78 D5 C5 E5 CD 4D 00 E1
D468 C1 23 79 E5 CD 4D 00 E1
D470 D1 23 7A D5 E5 CD 4D 00
D478 E1 D1 23 7B CD 4D 00 C9
D480 3A 94 EB FE 00 28 05 FE
D488 03 28 01 C9 3A B5 EB FE
D490 01 28 67 2A C7 F3 ED 5B
D498 B0 EB 19 3E 0D E5 CD 4D
D4A0 00 E1 23 3E FF CD 4D 00
D4A8 2A B0 EB 11 E0 01 ED 52
D4B0 28 4C 2A C7 F3 ED 5B B0
D4B8 EB 19 11 20 00 19 CD 4A
D4C0 00 FE 48 28 39 FE 4C 28
D4C8 35 FE FF 20 20 2A C7 F3
D4D0 ED 5B B0 EB 19 3E FF D5
D4D8 CD 4D 00 D1 21 20 00 19
D4E0 22 B0 EB ED 5B C7 F3 19
D4E8 3E 0D CD 4D 00 2A B0 EB
D4F0 2B 22 B0 EB 3E 01 32 B5
D4F8 EB C9 CD C2 D5 C9 CD 37
D500 D5 C9 2A C7 F3 ED 5B B0
D508 EB 19 E5 CD 4A 00 FE 00
D510 28 0C FE 01 28 08 FE 07
D518 28 04 FE 0B 20 05 3E 02
D520 32 AF EB E1 3E 11 E5 CD
D528 4D 00 E1 23 3E 13 CD 4D
D530 00 3E 00 32 B5 EB C9 2A
D538 C7 F3 ED 5B B0 EB 19 3E
D540 FF CD 4D 00 21 FF 00 22
D548 B0 EB C9 3A AE EB FE 00
D550 C2 46 D6 3A AD EB FE 01
D558 CA D3 D5 2A C7 F3 ED 5B
D560 AB EB 19 E5 2B 3E FF E5
D568 CD 4D 00 E1 11 20 00 19
D570 3E FF CD 4D 00 E1 3E 00
D578 E5 CD 4D 00 E1 11 20 00
D580 19 3E 01 E5 CD 4D 00 E1
D588 11 20 00 19 CD 4A 00 FE
D590 48 CA 40 D6 FE 4A CA 40
D598 D6 FE 25 CA 40 D6 FE FE
D5A0 CA 40 D6 3E 04 CD 41 01
D5A8 CB 57 20 0E 06 01 CB 5F
D5B0 20 02 06 81 78 32 AE EB
D5B8 18 09 CB 5F 20 05 3E 01
D5C0 32 AD EB 2A AB EB 11 DE
D5C8 00 ED 52 20 05 3E 01 32
D5D0 AF EB C9 2A C7 F3 ED 5B
D5D8 AB EB 19 11 21 00 19 E5
D5E0 CD 4A 00 E1 FE 24 28 58
D5E8 FE 34 28 4E FE 0D 28 50
D5F0 FE 11 28 4C 11 20 00 19
D5F8 CD 4A 00 FE 48 28 41 FE
D600 4A 28 3D FE FE 28 39 FE
D608 25 28 35 2A C7 F3 ED 5B
D610 AB EB 19 3E 04 E5 CD 4D
D618 00 E1 23 3E 06 E5 CD 4D
D620 00 E1 11 20 00 19 3E 07
D628 E5 CD 4D 00 E1 2B 3E 05
D630 CD 4D 00 2A AB EB 23 22
D638 AB EB 3E 00 32 AD EB C9
D640 3E 02 32 AF EB C9 3A AE
D648 EB FE 01 20 30 3C 32 AE
D650 EB 2A AB EB 11 20 00 ED
D658 52 22 AB EB ED 5B C7 F3
D660 19 3E 0A E5 CD 4D 00 E1
D668 11 20 00 19 3E 0B E5 CD
D670 4D 00 E1 11 20 00 19 3E
D678 FF CD 4D 00 C9 FE 02 20
D680 1D 3C 32 AE EB 2A AB EB
D688 ED 5B C7 F3 19 3E 00 E5
D690 CD 4D 00 E1 11 20 00 19
D698 3E 01 CD 4D 00 C9 FE 03
D6A0 20 1D 3C 32 AE EB 2A AB
D6A8 EB ED 5B C7 F3 19 3E 0A
D6B0 E5 CD 4D 00 E1 11 20 00
D6B8 19 3E 0B CD 4D 00 C9 FE
D6C0 04 20 1E 3E 00 32 AE EB
D6C8 2A AB EB E5 ED 5B C7 F3
D6D0 19 3E FF CD 4D 00 E1 11
D6D8 20 00 19 22 AB EB C3 4B
D6E0 D5 FE 81 20 6B 3C 32 AE
D6E8 EB 2A C7 F3 ED 5B AB EB
D6F0 19 11 21 00 19 CD 4A 00
D6F8 FE 24 CA 40 D6 FE 34 20
D700 07 2A AB EB 2B 22 AB EB
D708 2A AB EB ED 5B C7 F3 19
D710 E5 11 20 00 19 3E FF E5
D718 CD 4D 00 E1 23 3E FF CD
D720 4D 00 E1 3E 05 E5 CD 4D
D728 00 E1 23 3E 07 E5 CD 4D
D730 00 E1 11 20 00 ED 52 3E
D738 06 E5 CD 4D 00 E1 2B 3E
D740 04 CD 4D 00 2A AB EB 11
D748 20 00 ED 52 22 AB EB C9
D750 FE 84 CA EB D7 3C 32 AE
D758 EB 3A AD EB FE 01 CA 9A
D760 D7 2A AB EB ED 5B C7 F3
D768 19 3E FF E5 CD 4D 00 E1
D770 11 20 00 19 3E FF E5 CD
D778 4D 00 E1 23 3E 01 E5 CD
D780 4D 00 E1 11 20 00 ED 52
D788 3E 00 CD 4D 00 2A AB EB
D790 23 22 AB EB 3E 01 32 AD
D798 EB C9 2A AB EB ED 5B C7
D7A0 F3 19 3E 04 E5 CD 4D 00
D7A8 E1 23 3E 06 E5 CD 4D 00
D7B0 E1 11 20 00 19 3E 07 E5
D7B8 CD 4D 00 E1 2B 3E 05 CD
D7C0 4D 00 2A AB EB 23 22 AB
D7C8 EB ED 5B C7 F3 19 11 40
D7D0 00 19 CD 4A 00 FE 34 20
D7D8 0C 3E 00 32 AE EB 3E 04
D7E0 06 05 CD EC D8 3E 00 32
D7E8 AD EB C9 3E 00 32 AE EB
D7F0 2A AB EB ED 5B C7 F3 19
D7F8 E5 3E FF CD 4D 00 E1 2B
D800 3E FF E5 CD 4D 00 E1 11
D808 20 00 19 3E FF E5 CD 4D
D810 00 E1 23 3E FF CD 4D 00
D818 2A AB EB 11 20 00 19 22
D820 AB EB C3 4B D5 C9 2A AB
D828 EB ED 5B C7 F3 19 23 3E
D830 FF E5 CD 4D 00 E1 2B 2B
D838 3E FF E5 CD 4D 00 E1 11
D840 20 00 19 3E FF E5 CD 4D
D848 00 E1 23 23 3E FF E5 CD
D850 4D 00 E1 23 3E FF CD 4D
D858 00 2A AB EB E5 ED 5B C7
D860 F3 19 3E FF CD 4D 00 E1
D868 11 20 00 19 22 AB EB ED
D870 5B C7 F3 19 3E 00 E5 CD
D878 4D 00 E1 11 20 00 19 3E
D880 01 CD 4D 00 06 FF 3E FF
D888 3D 20 FD 10 F9 2A AB EB
D890 11 C0 02 ED 52 38 C2 3A
D898 9B EB 3D 32 9B EB C2 80
D8A0 D2 11 FA 00 3E 72 06 0C
D8A8 C5 D5 F5 2A C7 F3 19 CD
D8B0 4D 00 F1 3C D1 13 C1 10
D8B8 EF 06 1E 3E FF 0E FF 0D
D8C0 20 FD 3D 20 F8 10 F4 CD
D8C8 07 D3 3E 32 32 75 D3 C3
D8D0 5A D2 3A 94 EB 3C CB 97
D8D8 32 94 EB 3A 75 D3 3D 32
D8E0 75 D3 3E 03 06 05 CD EC
D8E8 D8 C3 80 D2 21 9D EB 16
D8F0 00 5F 19 E5 CD FE D8 E1
D8F8 10 F9 CD 07 D3 C9 7E 3C
D900 FE 0A 20 06 3E 00 77 2B
D908 18 F4 77 C9 3A 94 EB FE
D910 02 30 01 C9 01 B2 EB 21
D918 A9 01 CD 2A D9 21 71 01
D920 CD 2A D9 21 1A 01 CD 2A
D928 D9 C9 E5 ED 5B A4 EB ED
D930 52 E1 38 01 C9 0A 3C CB
D938 A7 02 03 FE 07 30 01 C9
D940 ED 5B C7 F3 19 FE 0F 06
D948 24 20 02 06 FF 78 CD 4D
D950 00 C9 00 00 00 00 00 00
E870 21 21 23 23 21 23 23 23
E878 21 23 23 21 23 21 23 23
E880 23 23 21 23 21 23 23 23
E888 23 21 23 23 21 23 23 23
```

DIVERTIMENTOS MATEMÁTICOS

- TIPOS DE QUEBRA-CABEÇA
- USANDO O COMPUTADOR
- SISTEMAS DE EQUAÇÕES
- TENTATIVA E ERRO
- TRUQUES MATEMÁTICOS

Desafie seu microcomputador a decifrar alguns quebra-cabeças matemáticos. Você também irá aprender técnicas de grande utilidade na resolução de sistemas de equações.



Ao chegar a este ponto de *INPUT*, você já teve contato com as principais técnicas de programação em BASIC. Dado um problema, será bem capaz de construir um programa para solucioná-lo (caso haja uma solução, é claro). Entretanto, a não ser que você tenha um *hobby* ou esteja trabalhando em algum projeto, são raras as suas oportunidades de mostrar o que sabe fazer em seu microcomputador.

Certos quebra-cabeças representam gostosos desafios para quem quer praticar programação ou simplesmente exercitar o raciocínio. A popularidade desse tipo de divertimento não é novidade. Os egípcios destacavam-se por sua habilidade em decifrar enigmas, assim como os gregos eram admirados pelos seus interessantes quebra-cabeças lógicos e matemáticos e seus paradoxos inexplicáveis. De fato, muitos problemas inicialmente tratados como simples recreação transformam-se em grandes descobertas científicas.

Encontrar uma solução elegante para um determinado problema geralmente traz uma satisfação muito grande. Mas, para os usuários de microcomputadores, a resolução de quebra-cabeças constitui, antes de tudo, um excelente exercício de programação — na verdade, bem mais eficiente que a análise e execução de programas prontos, uma vez que os estimula a escolher técnicas e aplicar idéias próprias.

QUEBRA-CABEÇAS NO COMPUTADOR

Existem diversos tipos de quebra-cabeça, e nem todos podem ser resolvidos no computador. Alguns exigem simplesmente intuição ou um pouco de raciocínio lógico. Um exemplo clássico seria o seguinte: um homem possui um lobo,



um carneiro e um repolho, e quer levá-los para o outro lado do rio em uma canoa. A canoa é muito pequena e ele só pode transportar um de cada vez. Mas não deve deixar sozinhos em qualquer das margens do rio nem o lobo e o carneiro, nem o carneiro e o repolho, pois algum não sobraria... Como chegar ao outro lado do rio com todos os seus bens?

Você pode resolver esse problema utilizando um computador (se quiser, escreva um programa), mas é muito mais rápido encontrar a sua solução com um lápis e um papel.

Os problemas mais adequados ao uso do computador são aqueles em que, dado um certo número de situações com os respectivos valores, pergunta-se sobre o valor de uma situação hipotética. A máquina também é útil quando a questão envolve cálculos aritméticos complexos ou geometria. Nesses casos, chega-se mais depressa ao resultado escrevendo um programa do que buscando a solução à mão.

Neste artigo, mostraremos como resolver três dos tipos mais comuns de quebra-cabeça. Antes de olhar as soluções, tente encontrá-las sozinho. Em seguida, examine os programas e veja como eles funcionam.

SIMPLIFICANDO O PROBLEMA

Se você não está acostumado a resolver quebra-cabeças, poderá ter algumas dificuldades em compreender o que é pedido em cada um deles, pois a maioria requer certa prática em extrair as informações essenciais de um texto razoavelmente confuso.

O primeiro tipo de problema, por exemplo, apresenta um texto de tamanho considerável que o torna muito mais complicado do que na realidade é. Vamos começar por uma situação bastante simples, que pode ser resolvida sem o auxílio do computador.

Em uma certa manhã de inverno, um grupo de amigos entra em uma lanchonete e toma três xícaras de café e duas de chá, pagando uma conta no valor de Cz$ 44,00. No dia seguinte, eles retornam ao mesmo lugar, mas tomam o dobro de xícaras de chá e um café a menos. Sabendo que desta vez a conta foi de Cz$ 48,00, qual é o preço de cada xícara de chá?

Se você conseguir remover todas as informações necessárias e colocar os dados principais sob a forma de variáveis, o problema consistirá na solução das seguintes equações: $3c + 2h = 44$ e $2c + 4h = 48$.

Estamos diante de um sistema de equações lineares: ambas devem ser resolvidas com os mesmo valores de **c** e **h**, que não estão elevados a nenhuma potência (3c↑2 − 2h = 44, por exemplo, não seria uma equação linear). Para se chegar a uma solução desse sistema são necessárias tantas equações quantas forem as variáveis. Neste caso, existem dois valores desconhecidos — **c** e **h** — e duas equações; logo, existe uma solução.

Uma maneira de resolver o problema seria isolar uma variável na segunda equação — $c = (48 - 4h)/2$ — e substituí-la na primeira, obtendo: 3 (48 − 4h)/2 2h = 44. Como resultado, teremos $h = 7$, ou seja, uma xícara de chá custa Cz$ 7,00.

Sistemas com duas equações são bem fáceis de resolver. Com três, também não são tão difíceis. Mas, a partir daí, você, certamente, irá precisar da ajuda do computador.

O VENDEDOR DE SELOS

Um negociante possui uma caixa de selos estrangeiros agrupados em seis tipos de envelope. Cada envelope, classificado de **A** a **F**, tem um preço diferente. Seis crianças, membros de um clube filatélico, gastaram todo o seu dinheiro do seguinte modo:

	A	B	C	D	E	F	Preço
Belinha	6	2	3	1	1	2	Cz$ 341
Nanda	14	11	0	1	2	1	Cz$ 469
Digo	0	1	3	6	4	3	Cz$ 598
Lu	5	3	5	2	1	1	Cz$ 376
Dudoca	12	1	4	4	3	2	Cz$ 587
Alcino	8	0	1	1	0	3	Cz$ 293

Qual é o preço de cada envelope?

Poderíamos resolver esse sistema eliminando uma variável por vez, mas, agindo assim, perderíamos muito tempo e qualquer erro aritmético inutilizaria o resultado.

Existem, porém, diversas maneiras de se resolver um sistema de equações — você mesmo seria capaz de inventar um método diferente. O que escolhemos para nosso programa é bem mais rápido e relativamente simples.



```
10 INPUT "DIGITE NUMERO DE LI
NHAS ";R
15 LET C=R+1
20 DIM A(R,C)· DIM B(R,C):
DIM A$(C-1,20)
30 FOR K=1 TO C 1
40 INPUT "NOMES DAS COLUNAS "
,A$(K)
50 NEXT K
60 FOR J=1 TO R
70 PRINT : PRINT "VALORES PAR
A A LINHA ";J
80 FOR K=1 TO C
90 INPUT A(J,K)
95 PRINT A(J,K);" ";
100 LET B(J,K)=A(J,K)
110 NEXT K  NEXT J
120 FOR L=1 TO R
130 GOSUB 230
140 GOSUB 280
150 NEXT L
160 CLS
170 FOR K=1 TO C-1· PRINT AT 1
,4*K-4;A$(K)· NEXT K
180 FOR J=1 TO R: FOR K=1 TO C
: PRINT AT 1+J,K*4-4;B(J,K)
190 NEXT K· NEXT J
200 PRINT "RESPOSTAS "
210 FOR K=1 TO C-1: PRINT AT 4
+R,K*4-4,A(K,C)· NEXT K
220 STOP
240 LET D=A(L,L)
250 FOR K=1 TO C
260 LET A(L,K)=A(L,K)/D
270 NEXT K· RETURN
290 FOR J=1 TO R
300 IF J=K THEN  NEXT J:
RETURN
310 LET F=A(J,L)
320 FOR K=1 TO C
330 LET A(J,K)=A(J,K)-F*A(L,K)
340 NEXT K: NEXT J: RETURN
```

```
10 CLS:INPUT"DIGITE NUMERO DE L
INHAS E COLUNAS ";R.C
20 DIM A(R,C),B(R,C),A$(C-1)
30 FOR K=1 TO C-1
40 INPUT"NOMES DAS COLUNAS ";A$
(K)
50 NEXT
60 FOR J=1 TO R
70 PRINT"VALORES PARA AS LINHAS
 ":J
80 FOR K=1 TO C
90 INPUT A(J,K)
100 B(J,K)=A(J,K)
110 NEXT K,J
120 FOR L=1 TO R
130 GOSUB 230
140 GOSUB 280
150 NEXT
160 CLS
170 FOR K=1 TO C-1:PRINT @4*K-3
,A$(K):NEXT
180 FOR J=1 TO R:FOR K=1 TO C:P
RINT @4*K-5+32*J,B(J,K)
190 NEXT K,J
200 PRINT"RESPOSTAS·"
210 FOR K=1 TO C-1:PRINT A$(K);
" = ",:PRINT USING"######.##";A
(K,C) NEXT
220 END
230 D=A(L,L)
250 FOR K=1 TO C
260 A(L,K)=A(L,K)/D
270 NEXT:RETURN
280 FOR J=1 TO R
300 IF J=L THEN NEXT:RETURN
```

```
310 F=A(J,L)
320 FOR K=1 TO C
330 A(J,K)=A(J,K)-F*A(L,K)
340 NEXT:NEXT:RETURN
```



```
10  HOME · INPUT "DIGITE O NUME
RO DE LINHAS E COLUNAS ";R,C
20  DIM A(R,C). DIM B(R,C): DIM
 A$(C - 1)
30  FOR K = 1 TO C - 1
40  INPUT "ENTRE COM OS NOMES P
ARA ASCOLUNAS ";A$(K)
50  NEXT K
60  FOR J = 1 TO R
70  PRINT : PRINT "INTRODUZA VA
LORES PARA A LINHA ";J
80  FOR K = 1 TO C
90  INPUT A(J,K)
100 B(J,K) = A(J,K)
110  NEXT K: NEXT J
120  FOR L = 1 TO R
130  GOSUB 240
140  GOSUB 290
150  NEXT L
160  HOME
170  FOR K = 1 TO C - 1: VTAB (
1): HTAB (K * 5)· PRINT A$(K):
NEXT K
180  FOR J = 1 TO R: FOR K = 1
TO C· HTAB (K * 5): VTAB (4 * J
): PRINT B(J,K)
190  NEXT K: NEXT J
200  VTAB (4 * J): HTAB (1): PR
INT "RESPOSTAS:-"
210  FOR K = 1 TO C - 1: VTAB (
4 * J + 4): HTAB (K * 5)· PRINT
  INT (A(K,C) * 100) / 100: NEX
T K
220  END
240 D = A(L,L)
250  FOR K = 1 TO C
260 A(L,K) = A(L,K) / D
270  NEXT K: RETURN
290  FOR J = 1 TO R
300  IF J = L THEN  NEXT J· RET
URN
310  LET F = A(J,L)
320  FOR K = 1 TO C
330 A(J,K) = A(J,K) - F * A(L,K
)
340  NEXT K: NEXT J: RETURN
```



```
10 CLS:INPUT "DIGITE O NUMERO D
E LINHAS E COLUNAS ";R,C
20 DIM A(R,C):DIM B(R,C):DIM A$
(C-1)
30 FOR K=1 TO C-1
40 INPUT "ENTRE COM OS NOMES PA
RA AS COLUNAS ";A$(K)
50 NEXT K
60 FOR J=1 TO R
70 PRINT:PRINT "INTRODUZA VALOR
ES PARA A LINHA ";J
80 FOR K=1 TO C
90 INPUT A(J,K)
100 B(J,K)=A(J,K)
110 NEXT K,J
120 FOR L=1 TO R
130 GOSUB 240
140 GOSUB 290
150 NEXT L
160 CLS
170 FOR K=1 TO C-1·LOCATE K*5+1
,1:PRINT A$(K)·NEXT K
180 FOR J=1 TO R·FOR K=1 TO C L
OCATE K*5,4*J:PRINT B(J,K)
190 NEXT K,J
200 LOCATE 1,4*J·PRINT "RESPOST
AS:-"
210 FOR K=1 TO C-1:LOCATE K*5,4
*J+4:PRINT INT(A(K,C)*100)/100
NEXT K
220 END
240 D=A(L,L)
250 FOR K=1 TO C
260 A(L,K)=A(L,K)/D
270 NEXT K·RETURN
290 FOR J=1 TO R
300 IF J=L THEN NEXT J RETURN
310 F=A(J,L)
320 FOR K=1 TO C
330 A(J,K)=A(J,K) F*A(L,K)
340 NEXT K,J·RETURN
```

A primeira parte do programa, da linha 10 à 110, permite que você introduza as informações, enquanto a segunda parte, da linha 120 à 150, chama as subrotinas de cálculo. A resposta será fornecida pela última parte, da linha 160 à 220.

Os valores são colocados em uma matriz **A(J,K)**, uma linha por vez. Neste programa, **J** se refere à linha da matriz e **K** corresponde à coluna. A matriz **A(J,K)** é copiada em outra idêntica — **B(JK)**. Conservamos, desse modo, a matriz original, que será impressa com a resposta, na linha 180.

O cálculo, feito linha por linha, é controlado pelo laço entre as linhas 170 e 210. Primeiro, a rotina entre as linhas 240 e 270 (230 e 270 no TRS-Color) seleciona o elemento de cada linha que pertence à diagonal da matriz (**A(1,1)**, **A(2,2)**, e assim por diante). Em seguida, divide cada um desses elementos pelo seu valor. Consequentemente, todos os elementos da diagonal da matriz tornam-se iguais a 1.

A próxima rotina executa a maior parte do trabalho. Embora tenha apenas seis linhas, é muito difícil entender o seu funcionamento.

Suponhamos que a rotina anterior já tenha feito **A(1,1) = 1**. O programa será então desviado para a linha 290 (280 no TRS-Color), com **L = 1**. A linha 300 não deixará que a linha 1 da matriz seja processada; assim, o laço partirá da linha 2. A linha 310 tomará o primeiro elemento da linha 2, colocando seu valor em **F**. Ainda em 2, a linha 320 irá selecionar cada coluna para que a linha 330 multiplique **F** pelo elemento da linha 1 da coluna em questão e subtraia esse resultado do elemento da linha 2 da coluna. O mesmo será feito com as linhas 3, 4, 5 e 6. Em seguida, o processo se repetirá com **L** = 2.

Ao final, os elementos pertencentes à diagonal da matriz continuarão iguais a 1, e os restantes, exceto os que se encontram na última coluna da direita, serão iguais a 0.

Será possível, então, ler diretamente os valores de **A**, **B**, **C** etc. na coluna não nula. Todos os microcomputadores, menos o Spectrum, apresentarão os resultados com duas casas decimais após a vírgula. Sem isso, uma resposta que seria 10 poderia aparecer como 9.999998 ou 10.000001.

O PRESENTE DE NATAL

Utilizando o programa anterior, você poderá resolver qualquer sistema de equações lineares que possua um número igual de equações e de variáveis. Muitos problemas, porém, incluem menos equações e apresentam incógnitas elevadas a algum expoente. Nesse caso, o melhor caminho para se chegar a uma resposta consiste no emprego da tentativa e erro. Geralmente, o próprio texto fornece alguma pista para encurtar seu trabalho.

Tente resolver este problema: no Natal, vovô Alberto, um excêntrico matemático, anunciou a seus dois jovens netinhos que cada um deles ganharia tantos pacotes quantos anos tivessem (contando apenas anos completos). Disse também que cada pacote conteria um número de envelopes correspondente à sua idade e que, em cada envelope, eles encontrariam, do mesmo modo, o valor em cruzados equivalente à idade. Por fim, comentou que, no ano seguinte, o mesmo presente lhe custaria Cz$ 500,00 a mais. Quantos anos têm seus netos?

Definindo a idade dos garotos como **A** e **B**, e o valor gasto no ano em questão como **M**, reduzimos o problema a duas equações.

A↑3 + B↑3 = M e
(A + 1) ↑ 3 + (B + 1)↑3 = M + 500

Temos, portanto, duas equações e três incógnitas. Sem recorrer à ajuda do computador, levaríamos um bom tempo experimentando valores de **A** e **B** que satisfizessem as duas equações. O computador trabalhará da mesma maneira — só que mais rapidamente e sem risco de cometer erros.

Antes de partir para os cálculos, devemos verificar se há no texto alguma informação que nos permita delimitar os

valores de **A** e **B**. A referência aos "jovens netinhos" sugere que os garotos não têm mais de catorze anos nem menos de três.

Como você poderá constatar, o programa para resolver o quebra-cabeça é bem simples.

```
10 FOR A=3 TO 14
20 FOR B=A TO 14
30 LET M=A^3+B^3
40 LET N=(A+1)^3+(B+1)^3
50 IF ABS (M+500-N)<.01 THEN
PRINT "A=";A,"B=";B
60 NEXT B
70 NEXT A
```

```
10 FOR A=3 TO 14
20 FOR B=A TO 14
30 LET M=A^3+B^3
40 LET N=(A+1)^3+(B+1)^3
50 IF ABS(M+500-N)<.01 THEN PRI
NT "A= ";A,"B= ";B
60 NEXT B
70 NEXT A
```

Embora o sistema possa apresentar mais de uma solução, obteremos apenas uma, devido à restrição introduzida nas linhas 10 e 20. Se o problema não especificasse, em sua formulação, "jovens netinhos", o laço **FOR...NEXT** seria bem mais extenso.

A função **ABS** da linha 50 evita que se cometa erros de arredondamento, checando se realmente **M + 500** é igual a **N**.

O método utilizado funciona para todos os problemas em que o número de incógnitas é maior que o número de equações. Outras situações podem exigir mais variáveis ou mais condições **IF...THEN**. Dependendo da complexidade do problema, o computador poderá levar alguns minutos ou então até horas para fornecer uma resposta, mas com certeza chegará a ela.

"MATEMÁGICA"

O terceiro grupo de quebra-cabeças que aparece em revistas especializadas em computador é o que envolve exclusivamente números. À primeira vista, parece simples questões aritméticas; mas, quando se tenta solucioná-las, logo se percebe a necessidade de recorrer à máquina.

Seguem-se alguns exemplos.

Existe um número formado de quatro dígitos que, quando invertido e multiplicado por outro número inteiro, retorna ao seu valor original. O problema consiste em descobrir se outros números apresentam essa mesma característica e, em caso afirmativo, apontar quais são eles.

Outro caso muito interessante é o do número 987654321. Ele é divisível por 17 e inclui todos os dígitos, de 1 a 9. Aqui, é preciso identificar um outro número que apresente essas mesmas propriedades.

Para solucionar ambos os problemas, devemos tratar o número como uma coleção de dígitos (e não como um valor numérico). Podemos dividir o número em unidades, dezenas, centenas etc. ou manipulá-lo como uma cadeia de caracteres, usando **RIGHT$**, **MID$** e **LEFT$** ou os comandos equivalentes no micro da linha Spectrum.

Resolveremos o problema do número de quatro dígitos empregando o primeiro método. Sendo tal número **ABCD**, chegaríamos a esta equação:

$$1000*\mathbf{A} + 100*\mathbf{B} + 10*\mathbf{C} + \mathbf{D} = \mathbf{X}*(1000*\mathbf{D} + 100*\mathbf{C} + 10*\mathbf{B} + \mathbf{A})$$

Como sempre, a maior dificuldade para encontrar as cinco incógnitas reside na delimitação do intervalo em que pode estar cada uma delas. **A** varia entre 1 e 9 — se fosse 0, teríamos um número de três dígitos. **B** e **C**, por sua vez, variam entre 0 e 9. O fator de multiplicação **X** não deve ser menor que 2 ou maior que 10/**D**, pois, nesse caso, a segunda metade da equação seria um número de cinco dígitos. Por essa mesma razão, **D** não pode ser maior que 4. Tendo delimitado esses intervalos, podemos escrever o programa para solucionar o problema.

```
10 FOR A=1 TO 9
20 FOR B=0 TO 9
30 FOR C=0 TO 9
40 FOR D=1 TO 4
50 FOR X=2 TO INT (9.9/D)
60 LET J=1000*A+100*B+10*C+D
70 LET K=1000*D+100*C+10*B+A
80 IF X*K=J THEN  PRINT J;"="
;X;"*";K
90 NEXT X: NEXT D: NEXT C:
NEXT B: NEXT A
```

```
10 FOR A=1 TO 9
20 FOR B=0 TO 9
30 FOR C=0 TO 9
40 FOR D=1 TO 4
50 FOR X=2 TO INT(9.9/D)
60 J=1000*A+100*B+10*C+D
70 K=1000*D+100*C+10*B+A
80 IF X*K=J THEN PRINT J;"=";X;
"*";K
90 NEXT X,D,C,B,A
```

Digite o programa e execute-o. Seja paciente: você precisará esperar um bom tempo para obter o resultado, pois o computador irá verificar todas as combinações possíveis.

Para solucionar o segundo problema, trataremos o número como se fosse uma cadeia de caracteres. O programa para o Spectrum é um pouco diferente dos programas destinados aos demais computadores, porque esse micro só manipula oito dígitos por vez, mas os princípios básicos são os mesmos.

```
 10 LET M=98765432
 20 LET M=M-27
 30 LET M$=STR$ M
 40 LET F=0
 50 FOR P=2 TO 9
 60 LET P$=STR$ P
 70 LET X=0: FOR K=1 TO 8: IF
 P$=M$(K TO K) THEN  LET X=K
 75 NEXT K
 80 IF X=0 THEN  LET F=F+1
 90 NEXT P: IF F=0 THEN  PRINT
 M$: STOP
100 GOTO 20
```

```
10 M=987654321
20 M=M-17
30 M$=STR$(M)
40 F=0
50 FOR P=1 TO 9
60 P$=CHR$(48+P)
70 X=INSTR(M$,P$)
80 IF X=0 THEN F=F+1
90 NEXT
100 IF F=0 THEN PRINT M$:END
110 GOTO 20
```

```
10 M = 987654321
20 M = M - 17
30 M$ =  STR$ (M)
40 F = 0
50  FOR P = 1 TO 9
60 P$ =  CHR$ (48 + P)
65  FOR Z = 1 TO  LEN (M$)
70  IF  MID$ (M$,Z,1) = P$ THEN
 90
80  NEXT Z:F = F + 1
90  NEXT P: IF F = 0 THEN  PRIN
T M$: END
100  GOTO 20
```

Fazendo uma contagem regressiva a partir de 987654321, o programa converte todo múltiplo de 17 em uma cadeia **M$**. As linhas 50 e 60 tratam cada um dos dígitos, de 1 a 9, como um caractere. A linha 70 verifica se o dígito está na cadeia **M$**; se não estiver, um indicador **F** é incrementado. A cadeia **M$** só será exibida se possuir todos os dígitos de 1 a 9.

MÚSICA, MICROS E MIDI

A interface MIDI oferece excelentes recursos para quem quer fazer música — amadorística ou profissionalmente. Veja como transformar seu computador em um sintetizador musical.

Embora seja um atributo relativamente recente dos micros, a capacidade de executar efeitos sonoros e musicais tem se tornado cada vez mais comum nos computadores pessoais de última geração. Muitos usuários com inclinações musicais chegam a escolher o computador em função de seus recursos sonoros. Os micros mais sofisticados possuem geradores de som de vários canais, embutidos, e permitem a programação dos mesmos através de comandos em BASIC.

Além de compor e executar músicas em um micro, o usuário tem, agora, a opção de conectar sua máquina a instrumentos musicais eletrônicos, como sintetizadores, órgãos e outros. Um padrão de interface, chamado MIDI (*Musical Instrument Digital Interface*), criado há poucos anos por um consórcio de indústrias eletrônicas japonesas e européias, está se firmando rapidamente, e promete abrir novas e numerosas possibilidades para o uso do computador em música. Examinaremos aqui esse periférico e suas características.

PRODUÇÃO DE SOM

A técnica de geração de efeitos sonoros e musicais em um micro evoluiu muitíssimo em relação aos "bipes" emitidos pelas primeiras marcas surgidas no mercado. Dos computadores cobertos por *INPUT*, o TRS-80 e o Apple são os que oferecem menos recursos para a geração de sons, exigindo programação elaborada em linguagem de máquina para a produção de algo mais complexo. O TK-2000 e o Spectrum têm comandos em BASIC, como o *SOUND*, que facilitam bastante a programação de efeitos sonoros, mas ainda em um nível mais simples. Já o TRS-Color e o MSX dispõem de recursos bem sofisticados, via comando **PLAY**. O MSX é o primeiro micro nacional com propriedades de sintetizador, pois permite o controle individual de algumas características da onda sonora, como a envoltória. Isso é possível graças a um circuito integrado específico para o controle de som.

Se você experimentou os programas para composição e execução de músicas, dados em artigos anteriores (como os das páginas 741 e 1009), certamente já tem uma idéia do que o seu computador pode (ou não) fazer. Seja através de recursos sonoros já embutidos na configuração básica do micro, seja através da adição de interfaces especiais (das quais existe uma grande variedade no mercado), as características mais importantes a observar em um bom sistema de geração sonora são:

- número de canais de saída sonora, ou vozes. Com apenas um canal, a música gerada é monofônica; o ideal é dispor de um mínimo de três vozes, para a sintetização de três instrumentos tocando simultaneamente;

- velocidade: se for muito baixa, não permite a execução de acordes múltiplos em cadência rápida;

- controle individual tanto de timbre como de intensidade;

- controle completo das características da onda sonora: bordo de ataque, bordo de fuga, envoltória etc. Esses recursos permitem sintetizar qualquer tipo de instrumento musical existente, ou até mesmo criar vozes para instrumentos que não existem;

- facilidade de programação: de preferência, deve ser possível utilizar comandos em BASIC;

- processador independente: permite que a melodia se inicie no ponto determinado pelo software principal, sendo executada independentemente do que a UCP estiver fazendo.

Apesar de toda a sofisticação das interfaces musicais de última geração, é preciso ter sempre em mente que a música gerada por um microcomputador nunca atinge os padrões de desempenho da música instrumental.

Devemos lembrar, ainda, que o teclado de um micro não corresponde ao teclado de um instrumento musical, como o piano, por exemplo, e é, na verdade, muito desajeitado para uso mais "sério". É aqui que entram em cena os instrumentos musicais digitais.

INSTRUMENTOS MUSICAIS

O desenvolvimento de novos instrumentos musicais nas últimas décadas é marcado por passagens semelhantes às que se observam na história das máquinas de calcular. Tradicionalmente, todos os instrumentos eram mecânicos: compunham-se de cordas, membranas, tubos etc. Aos poucos, a necessidade de obter intensidades sonoras maiores em *shows* destinados a grandes públicos (início da era do *rock*), assim como a intensa demanda criada pelas gravações, levou à eletrificação (ampliação eletrônica) de vários instrumentos musicais, como guitarras, órgãos e baterias. Finalmente, começaram a surgir instrumentos puramente eletrônicos — primeiro analógicos, e depois digitais — como o sintetizador.

As máquinas de calcular, por sua vez, evoluíram do ábaco para as calculadoras mecânicas movidas a manivela ou a motor elétrico, que, posteriormente, foram substituídas pelas modernas calculadoras eletrônicas. Como estas, os instrumentos musicais atuais estão recheados de *chips* integrados.

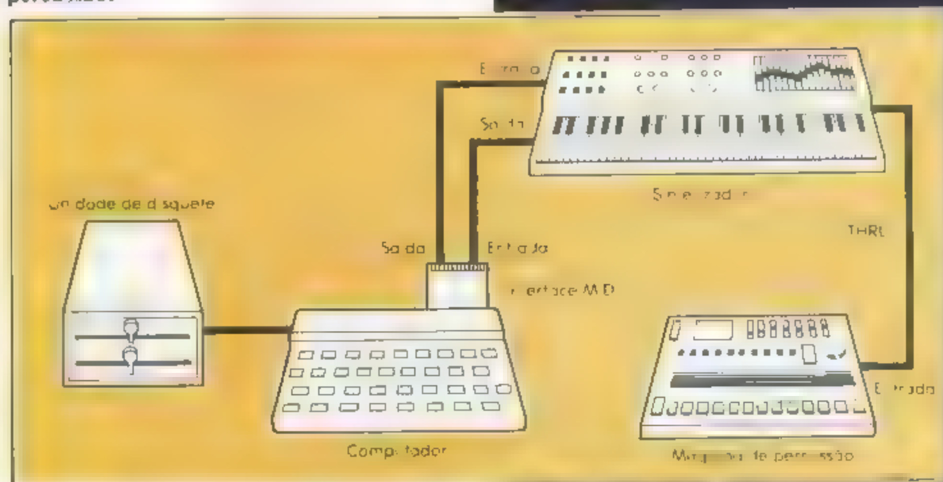
Os sintetizadores digitais são dispositivos sofisticadíssimos. Não se limitam à gama de notas e efeitos especiais oferecidos pelos micros (mesmo os mais modernos): dispõem de uma espantosa parafernália de recursos.

Um sintetizador comum, na faixa média de preço, permite a execução de acordes de até oito notas em um teclado como o de um piano. Quase todas as máquinas contam com um conjunto considerável de sons, envoltórias, timbres e ritmos especiais, possibilitando ao músico escolher instantaneamente entre um piano com eco, um violino ou um oboé, com acompanhamento de valsa, *bebop* ou batuque, em duas ou três vozes, com ou sem percussão.

Sons e instrumentos totalmente diferentes ou mesmo bizarros, efeitos de sirenas, explosões, gargalhadas, periquitos — ou o que vier à cabeça do executante — estão entre os recursos dos modelos de maior preço. Por meio de algumas teclas, eles nos dão acesso a uma orquestra completa.


RHYTHM COMPOSER TR-707

Uma rede MIDI ligando um computador, um sintetizador e uma máquina de percussão.





Em geral, o tipo mais comum de sintetizador utiliza o teclado de piano ou órgão como dispositivo de execução. Mas como o sintetizador é, na realidade, uma caixa cheia de circuitos eletrônicos, capazes de receber sinais das mais diversas origens, podemos recorrer a qualquer tipo de instrumento musical para gerá-los. Os mais utilizados, atualmente, são os de cordas (guitarra, contrabaixo etc) e os de percussão (bateria eletrônica).

Tomemos como exemplo as máquinas de percussão: elas podem funcionar isoladamente ou embutidas dentro de um sintetizador. A qualidade artificial da percussão eletrônica, evidente nos primeiros modelos, tornou-se praticamente imperceptível nos sintetizadores modernos. Muitos deles têm memória RAM que permite o armazenamento de seqüências complexas ou não rítmicas, para execução posterior.

Essa capacidade dos sintetizadores está promovendo uma verdadeira revolução na maneira de se fazer música. Até recentemente, a habilidade manual — a capacidade de mover os dedos com desenvoltura e rapidez sobre as cordas ou teclas ou de fazer soar um acorde ou batida no momento exato — era indispensável à execução musical. Com o advento do instrumento musical programável, isso está mudando.

O instrumento programável age no sentido de "liberar" o talento musical de uma dependência da destreza manual. Mas, com certeza, não o substitui: a sensibilidade para o ritmo e a capacidade de compor músicas mentalmente ou de obter um efeito desejado sempre serão necessárias.

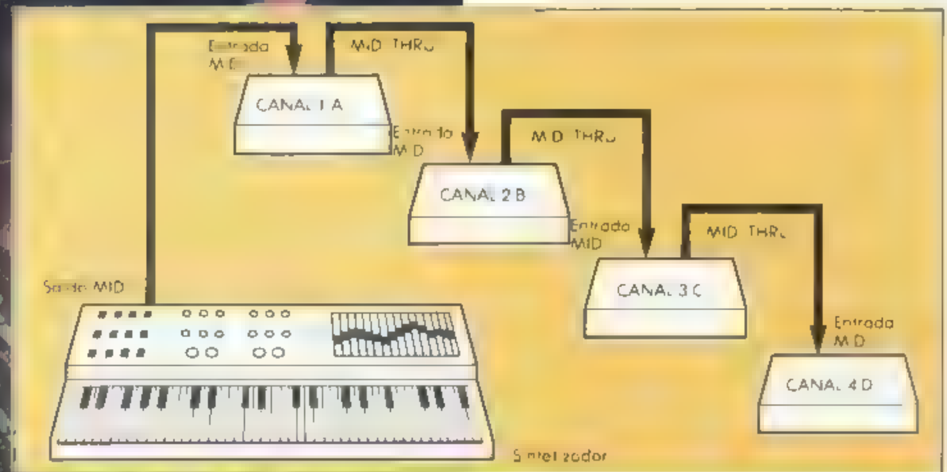
O sintetizador musical pode gerar sons de vários intrumentos musicais através do teclado. Se, por outro lado, você tiver a capacidade de programar e armazenar seqüências complexas de melodias, e tocá-las à vontade, até simultaneamente, seu "gênio musical" será enormemente ampliado.

Mas, antes que você se entusiasme, convém saber que os sintetizadores musicais programáveis são muito caros. E

é exatamente aqui que entra em cena a nova interface MIDI, salvando da frustração o amante da música que não tem um alto poder aquisitivo.

Com os computadores e os instrumentos musicais modernos utilizando a mesma tecnologia básica, tornou-se fácil interconectá-los e transmitir informações de um para o outro. É exatamente para isso que serve a MIDI.

Cada canal de informação de uma rede MIDI controla um instrumento.



O QUE É MIDI?

MIDI é um padrão industrial — da mesma forma que uma interface serial RS-232 ou uma paralela Centronics, usadas para conectar periféricos a um computador. Embora dirigida exclusivamente ao mundo da música, a MIDI desempenha um papel semelhante ao dessas interfaces, proporcionando um protocolo padronizado de transferência de informação entre o computador e um tipo especial de periférico: o instrumento musical digital. Isso assegura que qualquer instrumento (que seja compatível com o padrão MIDI) "entenda" o que o computador está ordenando.

Já se registraram outras tentativas de desenvolver um padrão de comunicação entre instrumentos musicais e computadores, mas nenhuma delas obteve uma grande aceitação. A MIDI, ao contrário, parece determinada ao sucesso: pelo menos os dois maiores fabricantes mundiais de sintetizadores e máquinas de percussão (Roland e Yamaha) já aderiram ao padrão MIDI. Além disso, vários microcomputadores lançados nos últimos anos, a começar do MSX, possibilitam a conexão à MIDI sem maiores dificuldades. Em conseqüência, prevê-se que esse padrão será universalmente adotado em pouco tempo.

Cada peça de equipamento compatível com a MIDI tem três soquetes de cinco pinos, do tipo DIN. Eles são rotulados de IN, OUT e THRU (alguns equipamentos MIDI mais antigos podem não ter este último tipo de conector). O soquete IN permite a recepção de sinais gerados em outro equipamento MIDI. THRU envia uma cópia idêntica dos sinais de entrada de um MIDI para ou-

tro, possibilitando a ligação de vários equipamentos "em cascata" (um equipamento que não tiver o soquete THRU é mais limitado, pois só pode ser conectado ao final de uma cadeia). OUT, finalmente, permite enviar sinais gerados em um equipamento para outro.

O padrão MIDI comporta até dezesseis canais simultâneos (paralelos) de informação. Cada canal controla um instrumento separado, mas a informação de vários instrumentos coexiste nos mesmos condutores elétricos (um artifício técnico denominado *multiplexação de alta velocidade*). Os instrumentos se encarregam de "sintonizar" a informação a eles encaminhada, da mesma maneira que um aparelho de televisão seleciona diversos canais.

UTILIZAÇÃO

Embora exista desde 1982, só recentemente a MIDI recebeu mais atenção do público. No Brasil, o interesse por essa interface surgiu com o advento dos primeiros microcomputadores compatíveis com a linha MSX. Entre suas aplicações mais frequentes, destaca-se o comando de um instrumento musical por outro. Interconectando um sintetizador e um órgão eletrônico, por exemplo, a MIDI permite que ambos sejam comandados de um único teclado.

A MIDI também possibilita a sincronização de dois instrumentos diferentes — como uma máquina de percussão e um piano elétrico —, assim como a ligação de um *sequenciador*. Esse dispositivo "memoriza" o que foi tocado na ordem correta, podendo repetir a execução de forma idêntica, tanto em tempo real quanto passo a passo. No primeiro caso, o sequenciador reproduz exatamente o que o músico tocou; já no segundo, ele executa, etapa por etapa, trechos da melodia, permitindo que o músico acrescente outras notas, de modo a preencher os segmentos de tempo.

MIDI E COMPUTADORES PESSOAIS

Como já mencionamos, o "casamento" entre computadores pessoais e a interface MIDI veio a público com o lançamento da linha MSX. Os fabricantes japoneses da linha MSX tinham um grande interesse nessa associação, uma vez que a maioria deles opera também com divisões altamente rentáveis de instrumentos musicais digitais. A Yamaha introduziu, com um custo muito baixo, um sintetizador semiprofissional completo: o modelo CX5M, que é um micro MSX com um sintetizador embutido e um teclado de piano. Essa máquina abre uma série de possibilidades interessantes para o músico, como a exibição na tela de vídeo das notações musicais de uma composição ou o uso do computador como um sequenciador de alta capacidade de memória, sem a necessidade de equipamentos extras.

Modelos diversos de microcomputadores — da linha MSX ou de outras linhas — podem ser utilizados com um equipamento MIDI acrescentando-se uma interface especial, ligada ao conector externo de expansão.

É provável que o preço de um computador completo ainda seja mais baixo que o de um sintetizador MIDI compatível. Mas, da mesma forma que aconteceu com outros periféricos, inicialmente muito caros — como impressoras, disquetes e monitores a cores —, o preço dos sintetizadores deverá diminuir bastante em um futuro próximo.

Antes mesmo que os preços comecem a cair, os usuários atraídos pela combinação microcomputador-instrumento musical devem ser informados sobre algumas peculiaridades desse periférico. Convém ter claro, em primeiro lugar, que a capacidade de geração sonora própria do micro não é usada quando se trabalha com a interface MIDI: o som é sempre gerado pelo sintetizador ou outro instrumento digital externo ao computador. Assim, não há vantagem em comprar um micro mais caro, dotado dos últimos recursos em matéria de geração musical. Mesmo a memória extra, disponível nos micros profissionais, não é tão importante, pois os computadores pessoais têm memória mais do que suficiente para o desempenho da função de seqüenciador.

Vale a pena observar, também, que a qualidade de som disponível não é limitada pelo meio de registro utilizado. Como o som é armazenado de forma digital (imune a ruídos), tanto faz usar uma fita cassete ou um disquete — a qualidade será sempre comparável à de um Compact Disc (CD). Em outras palavras, obtém-se na saída exatamente o que se colocou na entrada.

SOFTWARE

Uma vez que ligamos o computador a um ou mais equipamentos MIDI, é necessário um software especial para acionar o conjunto. A variedade de software é ainda restrita, e os preços são mais altos do que a média. A situação tenderá a mudar à medida que o uso da MIDI se popularizar.

Apesar das limitações, funções sofisticadas — como seqüenciamento, emulação de estúdios multicanais, composição e edição de melodias — são possibilitadas pelo software.

Mesmo quem não é capaz de tocar uma só nota em um instrumento musical normal, terá facilidade em compor temas musicais complexos no computador e enviá-los, em seguida, para execução. A composição pode ser armazenada digitalmente em fita ou disco, executada novamente, modificada etc. Já existem cartuchos de EPROM para micros compatíveis com o padrão MIDI, com conjuntos de músicas prontas, para autoacompanhamento ou, simplesmente, para se tocar no computador, usando-o como um sistema de alta-fidelidade.

Um pacote de software típico para a linha MIDI é o compositor musical, cujo funcionamento é exatamente igual ao de um editor de textos, só que trabalhando com a notação musical convencional, desenhada sobre a pauta. As suas funções mais comuns são inserção de músicas, apagamento, edição, listagem em vídeo ou impressora e execução musical em diversas cadências.

Um bom software coloca ainda à disposição do usuário uma série de funções de controle do sintetizador, tais como o comando independente de várias vozes (um sintetizador polifônico de qualidade deve ser capaz de tocar até dezesseis notas simultaneamente), seleção de ritmos, timbres ou sequências, mixagem etc. Se entre as características do sintetizador se incluir a capacidade de dividir o teclado, você poderá tocar um instrumento com a mão esquerda e outro com a direita.

PADRONIZAÇÃO

É possível enviar três tipos de informação através de uma interface MIDI: notas, mudanças de programa e mixagem de timbre. Atualmente, existe um conjunto padronizado de códigos MIDI que funciona como qualquer sintetizador compatível. Entretanto, esses códigos permitem apenas o controle das funções mais elementares. Funções especiais são controladas por sequência de códigos, que usualmente variam de instrumento para instrumento. Como resultado, um programa feito para um sintetizador MIDI pode não funcionar corretamente com outro. Além disso, para se programar orquestrações complicadas, é preciso estar familiarizado com a enorme gama de alternativas de controle. Provavelmente, também essas dificuldades serão atenuadas no futuro.

COMPUTADORES QUE OUVEM

Como já vimos, os periféricos de síntese de voz capacitam o micro a falar. O que talvez você não saiba é que as máquinas também podem entender a fala humana. Veja como.

No artigo *Computadores que Falam* (página 446), examinamos o funcionamento e as aplicações dos periféricos de síntese vocal, já disponíveis para a maioria dos microcomputadores. Como observamos, eles não são, em termos técnicos, muito complexos — existem modelos de preço acessível até para micros domésticos pequenos.

Se conferíssemos ao computador a habilidade inversa — ou seja, entender aquilo que falamos —, teríamos o "computador do futuro", que não precisaria de vídeo, nem de teclado para "conversar" conosco, humanos.

Mas isso não é tão fácil. Dotar a máquina da capacidade de reconhecer a fala é uma tarefa bem mais complicada do que fazê-la gerar a fala. Por essa razão, ainda não há sistemas capazes de um reconhecimento total da fala, nem mesmo nos mais sofisticados computadores de grande porte.

TIPOS DE SISTEMA

Não queremos dizer, com isso, que não se tenham desenvolvido sistemas com uma certa capacidade de reconhecimento da fala. Esses sistemas existem, inclusive em versões para microcomputadores pessoais, desde 1980/81.

Devemos identificar, portanto, o grau de *desempenho* de uma unidade de reconhecimento automático da fala (RAF). Dois critérios são utilizados. No primeiro deles, os sistemas são classificados em: sistemas de *reconhecimento da fala contínua* e sistemas de *reconhecimento de elocuções isoladas*.

Um sistema de reconhecimento da fala contínua deveria ser capaz de "entender" 90% das palavras emitidas no discurso natural — por exemplo, o pronunciamento de um deputado, uma conversa telefônica etc. E isso deveria ser feito em tempo real, ou seja, simultaneamente à fala. Sistemas com esse grau de desempenho ainda não existem. Os poucos sistemas de reconhecimento da fala contínua desenvolvidos até agora funcionam apenas em computadores imensos e custam muito caro. Além disso, eles são capazes de reconhecer um número muito restrito de palavras (o vocabulário é pequeno).

Já os sistemas de reconhecimento isolado, fabricados em uma grande variedade de modelos, são bem mais simples e baratos. Eles podem reconhecer com grande precisão um número restrito de palavras ou frases, enunciadas isoladamente e com clareza. O tamanho do vocabulário varia entre doze e trezentas palavras ou frases curtas, dependendo da sofisticação do sistema.

O segundo critério utilizado para a classificação dos sistemas RAF diz respeito à dependência do locutor.

Os sistemas *independentes do locutor*, como diz o nome, são capazes de reconhecer a fala de qualquer pessoa, em qualquer situação. Tais sistemas são tão raros quanto os de reconhecimento da fala contínua. Mais comuns são os sistemas cujo desempenho depende da pessoa que fala. Nesse caso, para que o sistema seja capaz de reconhecer a fala com um mínimo de precisão, a pessoa que vai usá-lo precisa "treinar" o computador, repetindo cada palavra ou frase do vocabulário um certo número de vezes. O programa analisa estatisticamente as repetições e as armazena na memória, para utilizá-las durante o processo de reconhecimento.

COMO FUNCIONA

O reconhecimento da fala requer o emprego de diversos algoritmos e processos. Inicialmente, a fala, captada por um microfone, é enviada ao computador e digitalizada por um conversor analógico digital. O conversor executa um processo de *amostragem*, ou seja, converte as ondas contínuas da fala em uma sequência de números digitais a cada 50/100 milissegundos.

Os dados, armazenados em um *buffer* (memória intermediária), são processados por um saftware especial, que analisa o conteúdo de freqüência de cada amostra (análise espectral). Isso resulta em uma matriz (tabela), em que cada coluna corresponde a uma amostra, e cada linha, a uma frequência. A matriz referente à palavra a ser identificada é comparada com as matrizes-padrão armazenadas internamente, uma para cada palavra do vocabulário. A palavra que, na comparação, acusar maior incidência é a palavra mais provável.

INTERFACES PARA MICROS

Existem, no exterior, diversas interfaces — em versões destinadas a microcomputadores pessoais ou profissionais — para reconhecimento automático da fala. Todos os modelos atualmente disponíveis empregam sistemas de reconhecimento de elocuções isoladas, dependente do locutor.

Para as linhas Apple, TRS-80 e TRS-Color, por exemplo, os sistemas Dragon e VoiceBox permitem o reconhecimento de 32 a trezentas palavras ou frases curtas, com 80% ou mais de precisão. Compõem-se de uma placa ou caixa de expansão, contendo um conversor AD, um conjunto de circuitos integrados especializados e software — parte do qual residente na memória ROM da unidade, parte em disquete.

Para utilizar o sistema, geralmente conecta-se o mesmo a uma porta de entrada serial do computador (RS-232C). Após carregar o software residente em disquete, o sistema pode ser operado com o auxílio de um microfone.

APLICAÇÕES

As aplicações do reconhecimento automático da fala são extremamente variadas, incluindo, por exemplo, o ensino de línguas, auxílio a pessoas inválidas, controle de aparelhos domésticos pela voz e até programação (existe uma versão "falada" do BASIC).

Existem também jogos muito divertidos para micros, como um jogo de pôquer, que utilizam voz sintética e reconhecimento da fala.

OS SEGREDOS DO TRS-80 (4)

Você sabe converter o sistema de posições de vídeo do seu micro em um sistema de coordenadas do tipo X e Y? Neste artigo, ensinaremos este e outros truques aos usuários do TRS-80.

Prosseguindo nossos estudos sobre a organização da tela dos micros compatíveis com a linha TRS-80, examinaremos como transformar o sistema de posições de vídeo, baseado na notação **PRINT @**, em um sistema de coordenadas verdadeiras, X e Y — recurso disponível para os usuários das linhas MSX, Apple, TK-2000 e Sinclair.

Relembrando o que já foi apresentado nesta série de artigos, a tela do TRS-80 é organizada em 1024 posições sequenciais, numeradas de 0 (canto superior esquerdo da tela) a 1023 (canto inferior direito). A numeração aumenta da esquerda para a direita em uma linha. O número-índice de cada posição na tela é o utilizado na expressão @ (arroba) de um **PRINT** posicional.

POSICIONAMENTO RELATIVO

A expressão @ corresponde a um posicionamento absoluto, ou seja, indica o local exato da tela onde deve começar uma impressão. Em diversas aplicações, porém, é interessante realizar posicionamentos relativos — como avançar para a próxima posição na linha de baixo, recuar uma posição na mesma linha, ir para o final da mesma linha e assim por diante.

Por meio de algumas expressões matemáticas, podemos realizar facilmente o posicionamento relativo, de modo a tratar a tela como um plano bidimensional. Essas expressões seguem um formato comum, em que se executa o cálculo de conversão e, ao mesmo tempo, se impõem os limites de variação mínima e máxima. Por exemplo: se a posição atual do cursor é **P** (um número inteiro entre 0 e 1023), podemos fazer com que ele avance uma posição para a direita por meio da expressão:

```
P%=P%-(P%+1<1024)
```

A notação intrínseca P% é utilizada para indicar que **P** é um número inteiro. Evitamos, assim, alguns erros de arredondamento nas expressões. A expressão acima combina, em uma única fórmula, uma adição e um teste:

```
P%=P%+1:IF P%>1023 THEN P%=1023
```

Ou seja, a posição P% sofre um incremento de 1, ao mesmo tempo em que é forçada a não exceder 1023.

Analisando a expressão completa, vemos que a subexpressão entre parênteses (P% + 1 < 1024) terá um valor igual a −1 se for verdadeira, isto é, se a próxima posição P% for menor que 1024. Nesse caso, ela será reduzida a P% = P% + 1, causando um incremento. Se a subexpressão for falsa, isto é, se P% + 1 for igual a 1024, seu valor será igual a 0 e P% receberá um incremento de 0 (ou seja, não será incrementado).

Em seguida, listamos as expressões mais úteis para posicionamento relativo. Estude-as cuidadosamente para entender como funcionam.

CONVERSÃO PARA COORDENADAS

O endereçamento linear da tela, tal qual é usado pelo interpretador BASIC do TRS-80, apresenta uma série de inconvenientes para a impressão de desenhos e textos que requerem o emprego de um sistema de coordenadas ortogonais (semelhante ao existente para a tela gráfica, com os comandos **SET, RESET** e **POINT**). Entretanto, não é difícil especificar um conjunto de equações que promova a transformação de um sistema de referência em outro.

Em primeiro lugar, precisamos adotar uma convenção referente à origem do sistema de coordenadas ortogonais. Se a origem deste corresponder ao canto superior esquerdo (X = 0 e Y = 0), e se X indicar a coluna, e Y, a linha da tela, estaremos usando a mesma convenção estipulada para os gráficos com **SET**. Neste caso, teremos:

1. Converter (X,Y) para uma posição @:

```
P% = 64*Y + X
```

2. Converter uma posição @ para (X,Y):

```
X = P%-INT(P%/64)*64
Y = INT(P%/64)
```

Se quisermos adotar a convenção cartesiana clássica (origem no canto inferior esquerdo), teremos:

1. Converter (X,Y) para uma posição @:

```
P% = 64*(15-Y) + X
```

2. Converter uma posição @ para (X,Y):

```
X = P%-INT(P%/64)*64
Y = 15-INT(P%/64)
```

O TECLADO DO TRS-80

O teclado dos micros compatíveis com a linha TRS-80 apresenta certas particularidades que, uma vez conhecidas, permitem a programação de alguns truques muito interessantes.

Da mesma forma que o vídeo, o teclado do TRS-80 tem uma área de memória, na RAM, reservada à armazenagem dos dados da matriz de entrada. Cada tecla, quando pressionada, envia um sinal por meio de dois condutores: um horizontal (percorre as fileiras de teclas) e outro vertical (percorre as colunas). Isso corresponde a um verdadeiro endereço da tecla, que tem seu correspondente em uma unidade da RAM.

Para compreender de que maneira o



PROTEJA O SEU PROGRAMA

Utilize o que você aprendeu sobre a desativação da tecla <BREAK> no TRS-80 para proteger o seu precioso programa em BASIC contra a curiosidade dos "piratas" de software.

A operação é simples: faça o computador carregar e executar automaticamente o programa. A primeira linha executável deve conter os comandos **POKE**, vistos neste artigo, que desativam a resposta à tecla **<BREAK>**. Essa providência impedirá que o programa seja interrompido e listado por meio do comando **LIST**.

Se o seu TRS-80 tiver o sistema operacional de disquetes (DOS), coloque o comando **BASIC NOME** (nome dado ao programa) no arquivo de auto-execução. Mas lembre-se de gravar NOME usando a opção de proteção contra listagem (**SAVE "NOME", P**).

teclado é processado e controlado pelo BASIC, precisamos conhecer uma outra área de memória RAM, que abrange diversos endereços — chamados coletivamente de *bloco de controle* do teclado. Como veremos em seguida, muitos desses endereços são úteis para nossos truques de programação.

A POSIÇÃO DO CURSOR

O BASIC do TRS-80 tem uma função intrínseca, chamada **POS(O)**, que informa a posição corrente do cursor dentro de uma linha. O número fornecido por essa função varia entre 0 e 63, e não informa a posição bidimensional na tela (posição @), que corresponde a um número de 0 a 1023.

A posição do cursor é armazenada no bloco de controle do teclado, nos bytes 16416 e 16417 (é um número de dezesseis bits, portanto). Para determinar a posição do cursor, basta usar um par de comandos **POKE**:

```
V = 256*PEEK(16417)+PEEK(16416)
```

Por meio dessa expressão, obtemos diretamente o endereço absoluto da memória de vídeo onde está o cursor. Para converter o valor resultante em uma posição **PRINT @**, basta subtrair o valor 15360, que corresponde ao primeiro endereço da memória de vídeo:

```
P = 256*PEEK(16417) + PEEK
(16416)-15360
```

Rodando o próximo programa, você terá uma demonstração desse cálculo. O programa preenche a tela com números inteiros, até chegar à posição 960, onde é chamada uma rotina de interrupção. Pressionando-se qualquer tecla, a tela é limpa, e a impressão continua a partir do último número exibido.

```
100 DEFINT N:N=0
110 CLS
120 N=N+1:PRINT N
130 NP=256*PEEK(16417)+PEEK(
16416) - 15360
140 IF NP<960 THEN 120
150 PRINT "APERTE <ENTER> P/
CONTINUAR"
160 IF INKEY$="" THEN 160 ELSE
110
```

REPETIÇÃO AUTOMÁTICA DE TECLAS

O teclado original do TRS-80, assim como o de seus compatíveis, não é repetitivo — ou seja, nada acontece se mantemos pressionada uma tecla. Trata-se de uma deficiência, pois seria conveniente, em muitas aplicações (jogos, por exemplo), que algumas teclas fossem auto-repetitivas.

Mas existe uma solução para esse problema: podemos saber se uma tecla está sendo pressionada ou não por intermédio do endereço 14591 do bloco de controle do teclado, que contém essa informação. Basta, portanto, consultar uma vez esse endereço — a cada repetição de um laço, por exemplo — e direcionar a lógica do programa de acordo com a informação obtida.

Para verificar a operacionalidade do endereço 14591 em seu micro, digite o seguinte programa:

```
10 PRINT PEEK(14591);:GOTO 10
```

Em seguida, digite **RUN** e experimente pressionar qualquer tecla. Note que, enquanto não se pressiona nenhuma tecla, o programa fica imprimindo zeros na tela. Um número maior que zero aparece quando se pressiona alguma tecla ou combinação de teclas.

O próximo programa ilustra com bastante clareza essa aplicação: uma espécie de "minhoquinha" é formada sobre a tela, quando se pressionam as quatro teclas com as flechas:

```
100 DEFINT A-Z:X=64:Y=24
110 T$ = CHR$(9) + CHR$(8) +
CHR$(91) + CHR$(10)
120 CLS
130 SET(X,Y):IF PEEK(14591)>0
THEN 160
140 C$=INKEY$:IF A$="" THEN 130
150 C=ASC(C$):GOTO 130
160 ON INSTR(T$,A$) GOSUB 200,
250, 300, 350
170 GOTO 130
200 X=X+1:IF X>127 THEN X=127
210 RETURN
250 X=X-1:IF X<1 THEN X=1
260 RETURN
300 Y=Y-1:IF Y<1 THEN Y=1
310 RETURN
400 Y=Y+1:IF Y>47 THEN Y=47
410 RETURN
```



O que é edição em tela completa?

Em um banco de dados, a ficha de entrada de informações é exibida na tela com todos os campos. *Edição em tela completa* é o recurso que permite ao usuário "passear" com o cursor por toda a tela, corrigindo e mudando as informações contidas nos vários campos, até ficar satisfeito com o resultado. Em seguida, por meio de uma tecla de controle, o conteúdo é armazenado na memória.

Conhecendo as funções do vídeo e do teclado do TRS-80, você não terá dificuldade em desenvolver um programa de edição em tela completa. Observe a seguinte estrutura:

- Inicialmente, a tela é limpa e os campos são exibidos na tela. Use comandos **PRINT @** para isso;
- em seguida, a sub-rotina de entrada de dados é chamada por **INKEY$**. Coloque nela vários **IF**, para detectar se alguma tecla de controle foi pressionada. Localize o cursor e efetue a operação solicitada;
- finalmente, tendo pressionado a tecla que assinala o fim da edição, leia (com **PEEK**) o conteúdo dos campos e transfira-o para a memória.

BLOQUEIO DA TECLA <BREAK>

Podemos desativar a tecla **<BREAK>** do TRS-80 Modelo 1 ou Modelo 3 (e dos microcomputadores compatíveis) se dermos os seguintes comandos (em modo direto, ou dentro de um programa):

```
POKE 16396,175:POKE 16397,201
```

Para ativar novamente, digite:

```
POKE 16396,201
```

Esse recurso é de grande utilidade quando desejamos impedir que um programa seja interrompido — intencional ou acidentalmente — pelo usuário.

PROGRAMANDO EM LOGO

A linguagem LOGO foi desenvolvida para facilitar o aprendizado da programação. Mas, se você pensa que ela se destina apenas a crianças, ficará surpreso com a riqueza de recursos.

Os computadores da década de 60 eram extremamente caros. A potência e a capacidade hoje disponíveis em um simples micro doméstico custavam alguns milhões de dólares nos computadores de grande porte de então, pois mesmo as máquinas maiores e mais sofisticadas contavam com um máximo de 144 Kbytes de memória RAM.

Assim, por razões de economia, as primeiras linguagens de programação foram projetadas para ocupar a menor quantidade de memória possível. Priorizava-se a facilidade de sua implementação no computador, e não a facilidade de uso para o programador.

Com o aparecimento dos microcomputadores, por volta de 1975, as linguagens dos antigos computadores tornaram-se bastante populares entre os usuários dos micros, pois essas máquinas, no começo, também dispunham de uma memória muito limitada. A maioria das pessoas aceitava como natural as dificuldades de aprendizado do BASIC e outras linguagens do gênero. "O que é simples para o computador também é simples para o programador", dizia-se.

A ESCOLHA DA LINGUAGEM

Quase todos os micros operam, originalmente, com um interpretador BASIC residente na memória ROM. E esta é a única linguagem utilizada pela maioria dos proprietários de micro.

Entretanto, não há motivo para que se limitem a uma única linguagem. O BASIC é apenas um programa em código de máquina, que pode tanto residir na memória quanto ser carregado de uma fita ou disquete. Nos micros profissionais, por exemplo, o BASIC é uma linguagem a mais, e precisa ser carregada quando se quer usá-la em programação. Isso significa que é perfeitamente possível carregar interpretadores de outras linguagens — ou seja, podemos mudar a linguagem que o computador "entende". Esses interpretadores são programas em código de máquina, capazes de entender o vocabulário e a sintaxe de uma linguagem de programação, e de traduzi-los em instruções executáveis pela UCP do micro.

No artigo da página 1288, vimos que, desde o aparecimento dos primeiros computadores, mais de duzentas diferentes linguagens de programação foram desenvolvidas. Essas linguagens servem aos mais variados propósitos. Muitas delas são extremamente especializadas, encontrando aplicação apenas em campos de pesquisa muito sofisticados. Outras são tão genéricas e fáceis de usar quanto o BASIC, e já estão disponíveis para microcomputadores.

Se você quiser utilizar uma determinada linguagem de programação no seu modelo de microcomputador, precisará, antes de mais nada, achar um programa interpretador ou compilador que possa ser executado nele. Linguagens alternativas são fornecidas em fitas ou discos, e devem ser carregadas na memória, antes de sua utilização. Algumas vezes, o interpretador também é encontrado em cartuchos removíveis da ROM (como os que existem para os micros TRS-Color e MSX), não precisando, evidentemente, ser carregado.

Os microcomputadores profissionais, com sistemas operacionais do tipo MS-DOS, CP/M ou UNIX, são os que dispõem de maior variedade de linguagens de programação. Já existe, porém, uma razoável oferta de linguagens alternativas para micros mais baratos — como os das linhas Spectrum, TRS-Color, Apple e MSX —, principalmente se puderem ser acoplados a disquetes.

NÍVEIS DE COMUNICAÇÃO

Uma linguagem de programação pode ser entendida tanto pelo usuário quanto pela máquina, constituindo uma espécie de mediação entre as linguagens de ambos — ou seja, entre a linguagem natural (o inglês, por exemplo) e a linguagem binária, empregada por todos os computadores existentes.

Quando a linguagem de programação está mais próxima da binária do que da natural, diz-se que é de *nível baixo*. O Assembler é um exemplo de linguagem desse tipo. Já as linguagens de *alto nível*, como o BASIC, são mais parecidas com a natural. As máquinas de quinta geração — a próxima geração de computadores — provavelmente irão utilizar linguagens superavançadas e poderão entender ordens dadas em linguagem natural ou seminatural. Mas as linguagens atuais — mesmo as mais avançadas, como o PROLOG — representam ainda um meio termo entre a facilidade de uso e a facilidade de implementação em um computador.

Além do BASIC, muito poucas linguagens de alto nível são utilizadas mais intensamente em microcomputadores pessoais: as principais são o LOGO e o PASCAL. Como veremos nos artigos desta série, essa linguagens são totalmente diferentes quanto a seu histórico e a seu campo de aplicação.

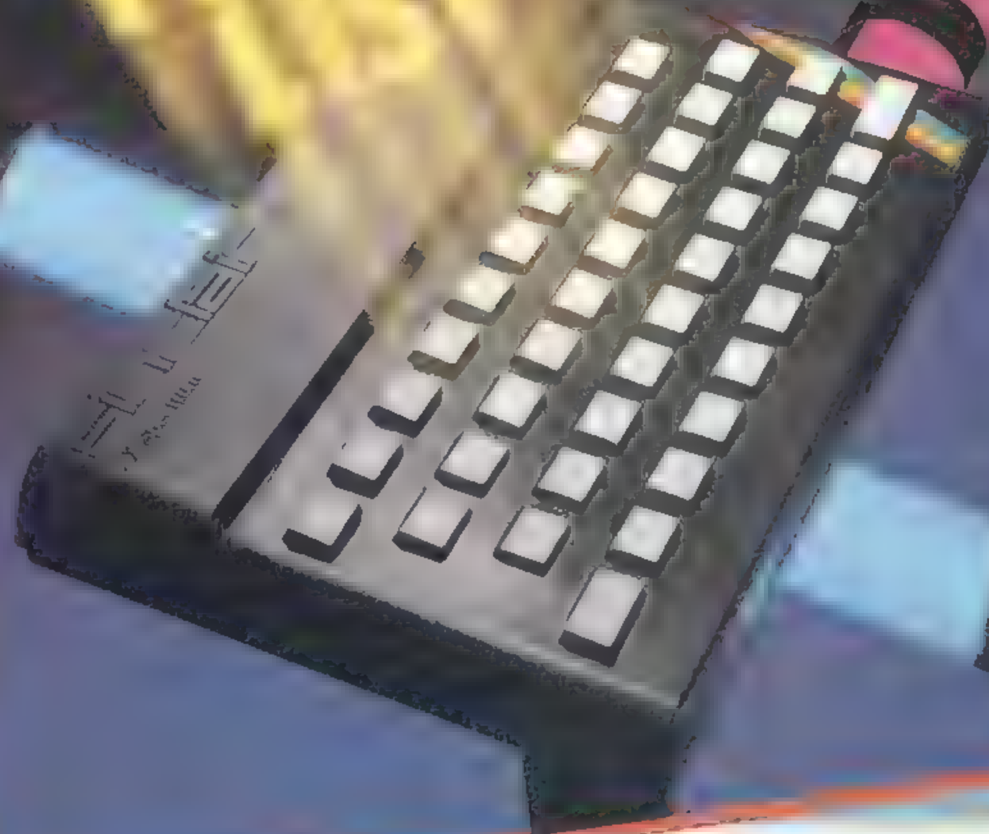
ORIGENS DO LOGO

Em 1967, um grupo de pesquisadores do famoso Massachusetts Institute of Technology (MIT), localizado na costa leste dos Estados Unidos, começou a explorar um caminho totalmente novo no desenvolvimento de linguagens para computadores. Esse grupo, liderado por Seymour Papert, um sul-africano expatriado, tinha como objetivo criar uma linguagem mais adaptada ao modo de funcionamento do intelecto humano do que ao processador central de um computador. Desse empenho nasceu o LOGO.

Papert tinha trabalhado com o famoso psicólogo e filósofo suíço Jean Piaget. As pesquisas de Piaget sobre o desenvolvimento da cognição evidenciavam que uma criança só é capaz de entender um conceito abstrato quando ele é apresentado por meio de exemplos concretos. As condições ideais de aprendizado estariam, assim, condicionadas ao envolvimento e experimentação pessoais. Essa abordagem influenciou fortemente a elaboração do LOGO.

Outra influência importante foi o trabalho de Marvin Minski, pesquisador do MIT e um dos pais da Inteligência Artificial — ciência que procura reproduzir aspectos da inteligência humana em computadores.

Os computadores não são inteligentes: sua capacidade se limita à obediência de instruções dadas com grande detalhe e precisão. A resolução de um pro-


LOGO
PASCAL
FORTH

blema intelectual envolve fatores tão complexos e variados, que programar uma máquina para uma operação desse tipo constitui uma tarefa especialmente desafiadora. Minski e seus colaboradores dedicaram-se a ela, buscando desenvolver uma linguagem de programação que facilitasse a simulação da capacidade de decisão e de resolução de problemas. Chegaram ao LISP (*LISt Processing*) — linguagem que passaram a utilizar desde o início dos anos 60.

A estrutura básica de dados do LISP é a *lista*, uma coleção de objetos elementares (chamados de *átomos*). Uma lista pode fazer parte de outras listas, o que torna mais fácil representar e processar dados simbólicos (não numéricos). Entretanto, o LISP é muito difícil de aprender.

O LOGO é essencialmente um dialeto do LISP — contém todas as estruturas e comandos que essa linguagem emprega para representar e processar dados simbólicos (palavras, listas etc.). Além disso, possui poderosos comandos gráficos, incluídos para facilitar seu uso por crianças pequenas.

DEMONSTRAÇÕES CONCRETAS

Papert e seus colegas estavam interessados em uma linguagem que permitisse iniciar as crianças no mundo da programação. Três áreas foram privilegiadas: desenho, música e robótica. Isto resultou da constatação de que as crianças se sentem mais motivadas a usar o computador quando a tarefa a realizar consiste em desenhar na tela, tocar música ou movimentar pequenos robôs com o auxílio do computador. Este último interesse levou Papert a criar uma "tartaruga" robótica.

Na época, não se encontravam monitores gráficos de vídeo a preços acessíveis. A tartaruga, um pequeno robô móvel sobre rodas, supria essa deficiência: carregava uma caneta em sua "barriga", e, sob o controle de comandos específicos do LOGO, podia abaixá-la e levantá-la. Assim, era capaz de desenhar figuras no chão, obedecendo às ordens da criança, que passava a relacionar os comandos da linguagem com os movimentos da tartaruga e com conceitos de orientação e geometria.

Quando surgiram os computadores pessoais, porém, a tartaruga-robô passou a correr risco de "extinção", pois era muito mais simples e barato imitar seu movimento na tela por meio de gráficos. Sua descendente direta é a tartaruga gráfica bidimensional — um simples cursor, em geral de forma triangular, que obedece às mesmas instruções de deslocamento usadas para movimentar a tartaruga-robô. Mas o simpático "animalzinho" não chegou a desaparecer. Ao contrário, tem se observado uma tendência a trazê-lo de volta às salas de aula. O pequeno robô é, sem dúvida, bem mais adequado ao ensino de crianças pequenas, que não entendem o formalismo da tartaruga gráfica. Além disso, é muito mais divertido!

OS ELEMENTOS DA LINGUAGEM

O que tem a ver a movimentação de uma tartaruga de brinquedo com programação de computadores e psicologia infantil? Papert vê no computador um importante veículo para a criatividade e a expressão de idéias, e acredita que se deve ensinar as crianças a dominá-lo, evitando que sejam dominadas por ele. A melhor maneira para que uma criança aprendesse a manipular computadores seria viver em uma "cultura computacional", assim como a melhor maneira de aprender italiano é viver na Itália. O LOGO — dando à criança os meios para usar os poderosos recursos disponíveis no computador — é a sua proposta para realizar isso. Como Papert procura demonstrar em seu livro *LOGO, Computadores e Educação* (*Mindstorms, Children, Computers and Powerful Ideas*, no original), essa linguagem funciona não só como uma ferramenta para a resolução de problemas, mas como um verdadeiro sistema de referência — denominado *micromundo* — para a introdução de novas idéias e conhecimentos.

Logo após o aparecimento do primeiro interpretador LOGO para computadores de grande porte, surgiram as primeiras versões para micros. Por sua capacidade gráfica superior, a linha Apple foi privilegiada com as versões iniciais mais poderosas da linguagem. Posteriormente, outros micros — entre os quais o Spectrum, o MSX, e o TRS-Color — ganharam interpretadores LOGO compatíveis com o padrão MIT.

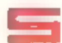
Como o campo de aplicação do LOGO é eminentemente educacional, os simbolismos da palavra escrita para a descrição da forma e funcionamento do nosso mundo tem grande importância. Assim, esta foi a primeira linguagem a ser amplamente traduzida para idiomas locais, inclusive o português. A partir de 1974, especialistas da Unicamp (Universidade Estadual de Campinas, no Estado de São Paulo) começaram a produzir o BRASLOGO, a versão brasileira do LOGO, que mais tarde foi adotada por diversas empresas fabricantes de microcomputadores, como a Itautec, a Gradiente e a Microdigital. Hoje, há versões dessa linguagem para as linhas Apple, MSX, TK-90X e CP-400, entre outras. Existem também diversos "pseudo-LOGO", assim chamados por serem versões muito reduzidas ou limitadas do LOGO original, do qual aproveitam apenas os recursos gráficos. Como o ColorLOGO, do TRS-Color I, esses pseudo-LOGOS não contam com os recursos de processamento de listas, funções matemáticas etc. da versão completa.

LOGO foi a primeira linguagem simbolica "amistosa", ou seja, facil de usar. Como ela se presta a aplicações educacionais, muitos julgam que se destina exclusivamente às crianças. Nada mais longe da verdade. Os recursos intrínsecos do LOGO (recursão verdadeira, processamento de listas etc.) colocam-no no mesmo *ranking* que o LISP, o SNOBOL, o PROLOG e outras linguagens empregadas em campos complexos, como Inteligência Artificial.



UM LOGO PARA SEU MICRO

A primeira providência a ser tomada quando se quer programar em LOGO é carregar o interpretador da linguagem. Cada linha de computador tem uma versão especial, pois ainda não existe uma padronização do LOGO.



O micro Spectrum dispõe de diversas versões do LOGO — e de pseudo-LOGO — em inglês. Estas, contudo, só podem ser obtidas no exterior.

No Brasil, a Microdigital lançou um LOGO em português para os computadores TK-90X e TK-95. Essa versão é carregada a partir de fita cassete e segue aproximadamente o padrão BRASLOGO (ou LOGO Itautec, desenvolvido pela Itautec em conjunto com a Universidade Estadual de Campinas).



Os modelos da linha TRS-Color só dispõem de um interpretador LOGO em cartucho ou disquete, com comandos e mensagens em inglês. Uma versão antiga, o ColorLOGO, é apenas um pseudo-LOGO gráfico, embora muito poderoso.



Esta é a linha que, internacionalmente, dispõe do maior número e variedade de interpretadores LOGO, todos eles carregados a partir de disquete. Em inglês, existem várias versões bastante eficazes, como o Terrapin LOGO, o Apple LOGO etc., que implementam o padrão MIT, além de terem comandos para geração de música, controle de tartaruga-robô e outros recursos.

Em português, a versão mais difundida é o MLOGO, desenvolvida por uma software-house paulista, a Micro-Arte. Essa versão, entretanto, não é compatível com o padrão BRASLOGO.





Os micros desta linha contam com vários interpretadores LOGO em inglês, em cartucho e em disquete. Todos aproveitam os excelentes recursos gráficos e sonoros do MSX para oferecer comandos especiais, que permitem o uso de sprites, tartarugas múltiplas etc.

No Brasil, há duas versões do LOGO em português. Uma delas, a da Gradiente, é apresentada em cartucho e seus recursos são bastante limitados. A outra — o HOTLOGO, da Sharp —, em cartucho e em disquete, é poderosíssima e segue o padrão BRASLOGO/Itautec.

As versões brasileiras aceitam palavras acentuadas segundo as regras da língua portuguesa.

PROGRAMANDO EM LOGO

Como não existe uma versão padronizada para o LOGO, todos os programas desta série serão dados em duas versões bem conhecidas: o padrão MIT, em inglês (listagens identificadas pelo símbolo do Apple), e o padrão BRASLOGO/Itautec, em português (listagens identificadas pelo símbolo do MSX).

No último artigo apresentaremos uma tabela de conversão de comandos para outras versões menos difundidas. Nas versões em português, a sintaxe é sempre a mesma; muda apenas a forma usada nas palavras-chave.

Quando se carrega o LOGO na memória (o que é desnecessário se a versão for em cartucho, pois, nesse caso, o programa se auto-executa), surge na tela uma mensagem inicial de "boas-vindas", que varia de acordo com o tipo de interpretador usado.

O sinal adotado pelo LOGO para avisar que está pronto a aceitar um comando é o ponto de interrogação. Ao lado desse sinal, aparece o cursor.

Inicialmente, vamos explorar os comandos gráficos do LOGO: colocaremos a tartaruga gráfica na tela e daremos comandos para movimentá-la, fazendo diversos desenhos. Para colocar a tartaruga na tela, digite este comando, e, depois, pressione a tecla **<ENTER>** ou **<RETURN>**:



```
SHOWTURTLE
```



```
TARTARUGA
```

O cursor aparece no centro da tela — sob a forma de um triângulo ou como uma tartaruga estilizada, conforme a versão do LOGO —, já pronto para desenhar. Por analogia com o robô mecânico, dizemos que a tartaruga está com a "caneta abaixada".

Dispomos de vários comandos de deslocamento da tartaruga. Ela pode andar para a frente ou para trás e virar para a direita ou para a esquerda. O deslocamento linear é medido em número de passos (*steps*) ou pontos gráficos, e a virada, em graus (ângulo). Por exemplo, para fazer a tartaruga andar trinta passos adiante, digite:



```
FORWARD 30
```



```
PARAFRENTE 30
```

Deixe um espaço entre a palavra de comando e o parâmetro (30, no caso). Sem esse espaço, o computador entenderá a linha como um comando único, que não é capaz de reconhecer, e mostrará uma mensagem de erro do tipo:



```
I DON'T KNOW HOW TO FORWARD30
```



```
VOCÊ AINDA NÃO ME ENSINOU
PARAFRENTE30
```

À medida que a tartaruga se desloca, traça uma linha reta na direção desejada, pois, como dissemos, ela já aparece na tela em modo de escrita.

Uma linha de comando pode ser corrigida com as teclas do cursor, antes de se digitar <ENTER> ou <RETURN>.

Para alterar a direção em que a tartaruga se movimenta, usa-se o comando que indica a direção e o ângulo da virada. Experimente:



```
RIGHT 30
```



```
PARADIREITA 30
```

Com isso, o símbolo da tartaruga apenas se inclina na direção desejada. Para que ela avance, digite:



```
FORWARD 50
```



```
PARAFRENTE 50
```

A linha traçada pela tartaruga forma um ângulo de 30 graus à direita da linha vertical anterior.

Comandos sucessivos podem ser dados na mesma linha. Experimente:



```
LEFT 30 BACK 50
```



```
PARAESQUERDA 30 PARATRAS 50
```

A linha de comando ordena, em suma, que a tartaruga retorne à orientação original (isto é, "olhando" para a parte superior da tela), por meio de uma virada de 30 graus, e que ande para trás cinqüenta passos.

ABREVIATURAS

Para facilitar a digitação, podemos recorrer à forma abreviada de muitos comandos da linguagem LOGO. Veja abaixo as abreviaturas para os comandos que foram mostrados até agora:



```
SHOWTURTLE        ST
FORWARD           FD
BACK              BK
LEFT              LT
RIGHT             RT
```



```
TARTARUGA         TAT
PARAFRENTE        PF
PARATRAS          PT
PARAESQUERDA      PE
PARADIREITA       PD
```

MAIS COMANDOS

O LOGO possui uma grande variedade de comandos (também chamados *primitivos*). Vejamos mais alguns deles, relacionados à elaboração de gráficos:



HOME - Leva a tartaruga de volta ao centro da tela, sem apagá-la.

CLEARSCREEN - Apaga o conteúdo da tela e leva a tartaruga de volta ao centro (abreviação: **CS**).

PENUP - "Desliga" o modo de escrita da tartaruga: ao se movimentar, ela não deixará nenhum traço (abreviação: **PU**).

PENDOWN - "Liga" o modo de escrita da tartaruga (abreviação: **PD**).

PENCOLOR - Muda a cor do traço e das letras. É seguido de um número entre 0 e 6, que indica o código da cor (abreviação: **PC**).

Digite o programa que se segue para ter um exemplo da utilização dos comandos aprendidos até o momento. Nunca se esqueça de pressionar a tecla <ENTER> após cada linha de comando.

```
SHOWTURTLE
LEFT 45
FORWARD 71
RIGHT 135
PENUP
FORWARD 50
PENDOWN
LEFT 45
BACK 71
PENUP
HOME
PENDOWN
```

O comando **CLEARSCREEN** se encarregará de limpar toda a tela, caso você queira realizar outras experiências.



PARACENTRO - Leva a tartaruga de volta ao centro da tela, porém sem apagá-la (abreviação: **PC**).

APAGUEDESENHO - Apaga o conteúdo da tela e leva a tartaruga de volta ao centro (abreviação: **AD**).

USENADA - "Desliga" o modo de escrita da tartaruga: ao se mo-

vimentar, ela não deixará nenhum traço (abreviação: UN).

USELÁPIS - "Liga" o modo de escrita da tartaruga (abreviação: UL).

USEBORRACHA - Apaga todos os traços por onde passar a tartaruga (abreviação: **UB**).

MUDECL - Muda a cor do traço e das letras. Deve ser seguido de um número entre 0 e 7, que indica o código da cor.

Eis aqui um exemplo para ilustrar o uso dos comandos aprendidos até agora. Não se esqueça de pressionar a tecla **<ENTER>** após cada linha de comando.

```
TARTARUGA
PARAESQUERDA 45
PARAFRENTE 71
PARADIREITA 35
USENADA
PARAFRENTE 50
USELAPIS
PARAESQUERDA 45
PARATRAS 71
USENADA
PARACENTRO
USELAPIS
```

MONTE UM PROCEDIMENTO

Todos os comandos examinados foram dados em modo direto, ou "imediato", que equivale ao modo direto do BASIC — ou seja, o comando é executado imediatamente pelo interpretador, e logo esquecido. Entretanto, existe um outro modo em LOGO: o modo programado, ou modalidade procedimento.

Nessa modalidade, atribui-se um nome a um conjunto de comandos, que passam, então, a ser conhecidos como um *procedimento*. O nome que for dado ao procedimento torna-se parte integrante do vocabulário do LOGO. Diz-se, por essa razão, que o LOGO pertence à família das linguagens extensíveis.

Quando se digita o nome de um procedimento registrado, o interpretador executa todos os comandos contidos no mesmo, em seqüência. Assim, pode-se dizer que o procedimento também é um programa. Mas, ao contrário de um programa em linguagem BASIC, dispensa a numeração das linhas.

Para iniciar a entrada de um novo procedimento, use o comando:



```
TO NOME
```



```
APRENDA NOME
```

Na versão em português, o comando **APRENDA** pode aparecer na sua forma abreviada, ou seja, **AP**.

O NOME do procedimento pode ser qualquer um e ter qualquer tamanho. Só não deve conter espaços em brancos ou coincidir com o nome dos primitivos da linguagem LOGO.

Você poderá entender melhor de que maneira funciona um procedimento por meio de um exemplo. Vamos definir um procedimento chamado ZIGZAG. Logo que digitamos o comando inicial — **TO ZIGZAG**, em inglês, ou **APRENDA ZIGZAG**, em português —, tudo o que estava na tela desaparece, e entra em ação um pequeno *editor de textos*, que permite a entrada dos comandos que constituirão o procedimento. O sinal de prontidão muda de ? (ponto de interrogação) para > (sinal de maior).

Para finalizar o procedimento, pressiona-se a tecla **END** (para as versões em inglês) ou **FIM** (versões em português). O sinal de ? aparece novamente, indicando que voltamos ao modo direto. Em alguns computadores, como o Apple, por exemplo, é necessário pressionar as teclas **<CTRL>** e **<C>**



```
TO ZIGZAG
FORWARD 20 LEFT 150
FORWARD 20 RIGHT 150
FORWARD 20 LEFT 150
FORWARD 20 RIGHT 150
FORWARD 20 LEFT 150
FORWARD 20 RIGHT 150
END
```



```
APRENDA ZIGZAG
PARAFRENTE 20 PARAESQUERDA 150
PARAFRENTE 20 PARADIREITA 150
PARAFRENTE 20 PARAESQUERDA 150
PARAFRENTE 20 PARADIREITA 150
PARAFRENTE 20 PARAESQUERDA 150
PARAFRENTE 20 PARADIREITA 150
FIM
```

ZIGZAG é, agora, um novo comando do LOGO. Se você digitar esse comando, a tartaruga executará os comandos gráficos do programa anterior e traçará um ziguezague na tela.

Como você deve ter observado, o programa compõe-se de três repetições da mesma sequência de comandos, uma para cada "perna" do ziguezague. Mas o LOGO tem um comando que permite especificar o número de repetições (semelhante ao **FOR ... NEXT** do BASIC). Utilizando-o, e abreviando os comandos gráficos, nosso programa ficará assim:



```
TO ZIGZAG
REPEAT 3 [FD 20 LT 150 FD 20 RT
150]
END
```



```
APRENDA ZIGZAG
REPITA 3 [PF 20 PE 150 PF 20 PD
150]
FIM
```

A rotina a ser repetida, delimitada pelos sinais [e] (não use parênteses), é precedida pelo comando de repetição e pelo número de vezes que ela deve ser repetida. O comando de repetição é muito útil para a obtenção de figuras geométricas regulares. Podemos traçar um semicírculo, por exemplo, com o seguinte comando.

MICRO DICAS

CONVERSÃO PARA O MLOGO

Um dos interpretadores LOGO mais usados no Brasil é o desenvolvido pela Micro-Arte, de São Paulo, para computadores da linha Apple. Como nessa versão os comandos primitivos do Apple LOGO foram traduzidos para o português em desacordo com o padrão BRASLOGO, apresentamos abaixo os comandos do MLOGO correspondentes aos utilizados neste artigo. A sintaxe permanece a mesma.

BRASLOGO	MLOGO
TARTARUGA	MOSTRET
PARACENTRO	CENTRO
APAGUEDESENHO	LIMPETELA
PARAFRENTE	FRENTE
PARATRÁS	VOLTE
PARADIREITA	DIREITA
PARAESQUERDA	ESQUERDA
USELÁPIS	COLORIDO
USENADA	SEMCOR
USEBORRACHA	
MUDECL	COR
REPITA	REPITA
APRENDA	APRENDA
FIM	FIM
ADEUS	ADEUS



```
REPEAT 180 [FORWARD 1 RIGHT 1]
```



```
REPITA 180 [PARAFRENTE 1
PARADIREITA 1]
```

UMA PISTA DE CORRIDA

Agora que temos todos os "ingredientes" para montar um programa mais complexo, vamos tentar desenhar uma pista de corrida na tela. Para isso, dividiremos o problema em várias partes, ou subtarefas, que depois serão chamadas na ordem correta.

Começando com a parte interna da pista, definiremos um procedimento que trace a curva e, em seguida, a reta:



```
TO LADOINTERNO
FORWARD 100
REPEAT 180 [FORWARD 1 RIGHT 1]
END
```



```
AP LADOINTERNO
PF 100
REPITA 180 [PF 1 PD 1]
FIM
```

A combinação de duas curvas e duas retas nos fornecerá a parte interna da pista de corrida:



```
TO PISTAINTERNA
LADOINTERNO LADOINTERNO
END
```



```
AP PISTAINTERNA
LADOINTERNO LADOINTERNO
FIM
```

A parte externa da pista requer uma curva maior. Para obtê-la, aumentamos o passo da tartaruga:



```
TO LADOEXTERNO
FORWARD 100
REPEAT 90 [FORWARD 3 RIGHT 2]
END
```

e.

```
TO PISTAEXTERNA
LADOEXTERNO LADOEXTERNO
END
```



```
AP LADOEXTERNO
PF 100
REPITA 90 [PF 3 PD 2]
FIM
```

e:

```
AP PISTAEXTERNA
LADOEXTERNO LADOEXTERNO
FIM
```

Para começar a pista em um ponto adequado da tela, definimos o procedimento chamado INICIO:



```
TO INICIO
PENUP
LEFT 40 FORWARD 110 RIGHT 130
PENDOWN
END
```



```
AP INICIO
USENADA
PE 40 PF 110 PD 130
USELAPIS
FIM
```

Um outro procedimento, PARTIDA, moverá a tartaruga para onde a linha de partida e o início da parte interna da pista devem ser desenhados:



```
TO PARTIDA
RIGHT 90 FORWARD 30 LEFT 90
END
```



```
AP PARTIDA
PD 90 PF 30 PE 90
FIM
```

Finalmente, para traçar a pista toda de uma vez, definiremos um procedimento chamado PISTA.



```
TO PISTA
INICIO
PISTAEXTERNA
PARTIDA
PISTAINTERNA
END
```



Como posso traduzir os comandos de meu LOGO que estão em inglês?

Nada mais fácil. Como o LOGO é uma linguagem extensível, podemos desenvolver vários procedimentos simples, que obedecem a um comando em português e chamam o primitivo equivalente em inglês. Se você resolver, por exemplo, trabalhar com um BRASLOGO dentro do LOGO original, defina:

```
TO PARAFRENTE :D
FORWARD D:
END

TO PARATRAS :D
BACK :D
END

TO PARADIREITA :A
RIGHT :A
END

TO PARAESQUERDA :A
LEFT :A
END

TO PARACENTRO
HOME
END

TO REPITA :N :LIST
REPEAT :N :LIST
END
```

e assim por diante. Em seguida, grave esse conjunto de procedimentos com **SAVE "BRASLOGO**, e, quando quiser programar em português, carregue-o com **LOAD "BRASLOGO**



```
AP PISTA
INICIO
PISTAEXTERNA
PARTIDA
PISTAINTERNA
FIM
```

Simples, não é?

Todos os programas em linguagem LOGO são elaborados dessa maneira. A divisão do programa principal em uma série de "tijolos" de construção torna bem mais fácil tanto a programação como a própria execução de testes. No próximo artigo, veremos como montar programas de maior complexidade usando essa mesma técnica.

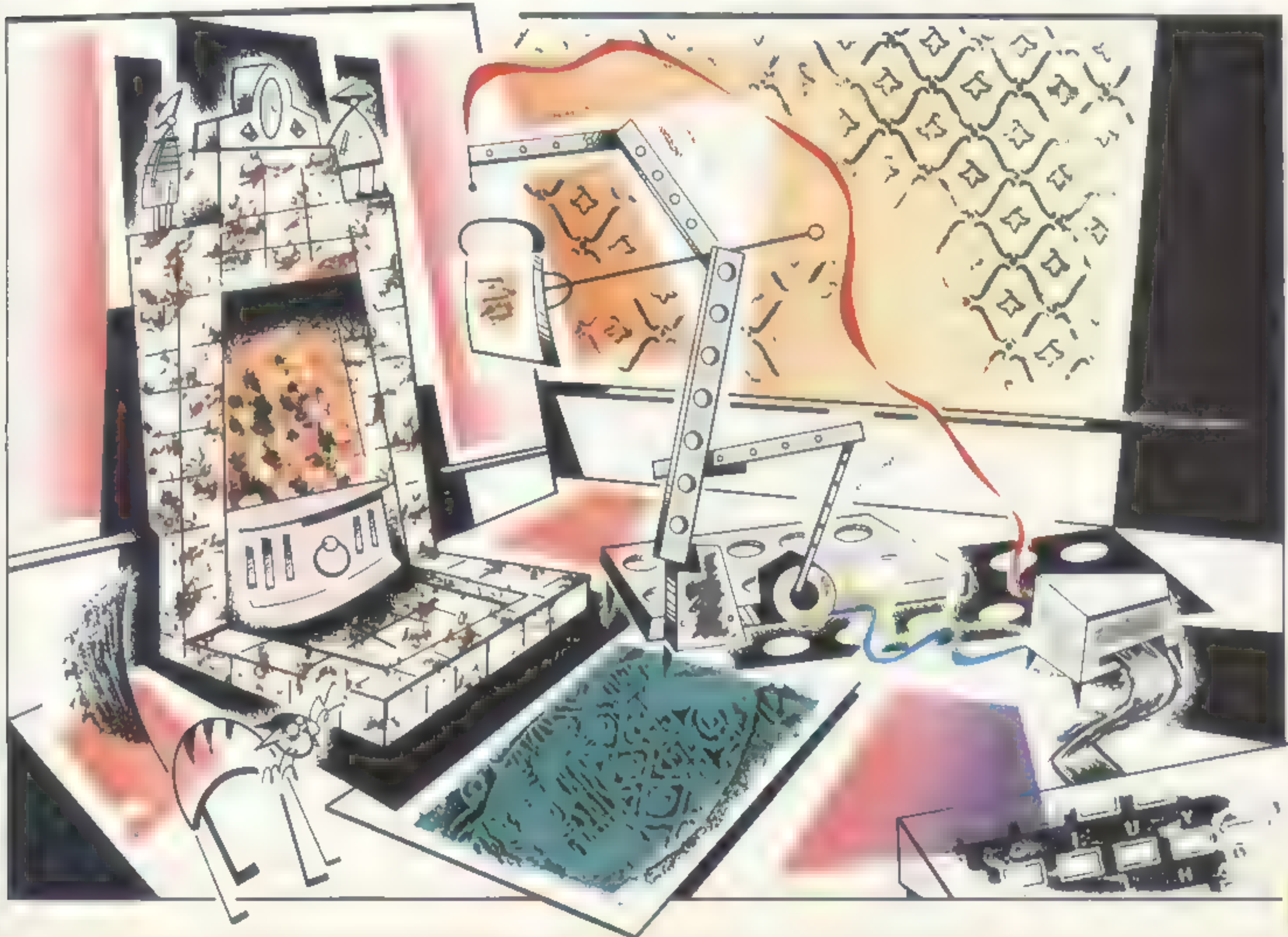
CONTROLE POR COMPUTADOR

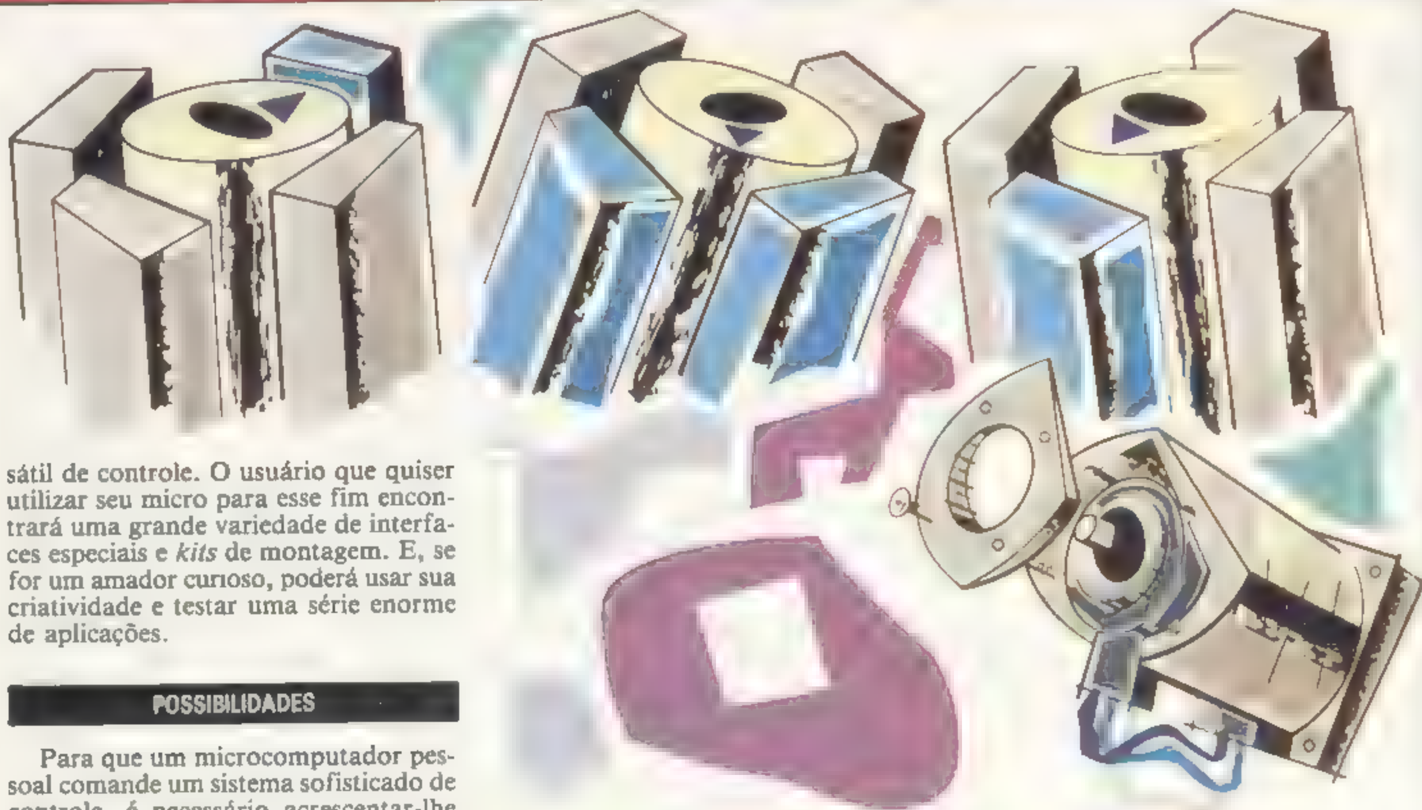
Como um computador pode receber informação do mundo exterior, analisá-la e usar os dados para operar dispositivos de diversos sistemas? Explicamos aqui todo o processo.

A microeletrônica provocou uma verdadeira revolução nos sistemas de controle de uma grande variedade de máquinas de uso doméstico e industrial. Dezenas de relés, cronômetros mecânicos e motores elétricos, usados antigamente para controlar aparelhos domésticos, foram substituídos por uma pequena caixa. Os circuitos e componentes microeletrônicos nela contidos realizam a mesma função de modo muito mais eficiente e sem falhas e avarias. Na indústria, inúmeros tipos de máquinas, como tornos e cortadores computadorizados e robôs de montagem, transformaram o dia-a-dia das fábricas.

Existem quatro tipos básicos de sistema microeletrônico de controle: circuitos lógicos discretos, circuitos integrados especiais, circuitos integrados genéricos e microprocessadores. Dentre todos, os microprocessadores foram os mais revolucionários, pois, além de ampliar o alcance das aplicações, eles baratearam o custo dos sistemas de controle, através do conceito de *controle por software*. Um microprocessador tem uma infinidade de aplicações, bastando reprogramá-lo. Os outros três tipos de sistema, ao contrário, não podem ser mudados, a não ser que se façam modificações no hardware.

Diversos tipos de equipamentos e aparelhos contêm microprocessadores embutidos, que funcionam com base em softwares específicos para cada tarefa de controle. Evidentemente, apenas o microprocessador não é suficiente para essa aplicação: são necessários *chips* auxiliares adicionais (circuitos integrados), conectados ao processador, para que ele possa exercer alguma função útil. Assim, todo proprietário de um computador pessoal tem em suas mãos o coração de um sistema potencialmente ver-





sátil de controle. O usuário que quiser utilizar seu micro para esse fim encontrará uma grande variedade de interfaces especiais e *kits* de montagem. E, se for um amador curioso, poderá usar sua criatividade e testar uma série enorme de aplicações.

POSSIBILIDADES

Para que um microcomputador pessoal comande um sistema sofisticado de controle, é necessário acrescentar-lhe uma série de dispositivos.

Uma aplicação típica de controle sempre requer uma ou mais entradas e saídas especiais, operadas de maneira bem diferente das entradas e saídas mais comuns em um microcomputador. Um mecanismo de controle de temperatura, por exemplo, precisa ser ativado quando a temperatura atinge um determinado grau. O computador recolhe essa informação por intermédio de um *sensor*, que gera um sinal relativo ao fenômeno físico ou químico que está sendo detectado e o envia ao computador. Para agir sobre o mundo exterior no sentido de combater esse aumento de temperatura, o computador deverá estar ligado a um *atuador* — no caso, um condicionador de ar ou um ventilador.

No artigo sobre robôs (página 1284), descrevemos operações desse tipo. Os kits de braços robóticos, ali mencionados, são um exemplo de sistemas de controle especializados.

Neste artigo, examinamos os aspectos gerais do emprego de computadores pessoais no controle de dispositivos externos. Nossa discussão será centrada nos três principais elementos de um sistema de controle: sensores, atuadores e conectores. Antes, porém, analisaremos as possibilidades de controle.

Convém explorar mais detalhadamente quatro áreas:

- temporização e seqüenciamento
- regulação
- redução de dados
- interfaces homem-máquina

TEMPORIZAÇÃO E SEQÜENCIAMENTO

Sistemas de temporização e seqüenciamento fazem parte do nosso cotidiano: são usados nos programadores de máquinas de lavar, máquinas de costura, aquecimento central etc. O computador que administra muitos desses sistemas é bem mais compacto e confiável do que os sistemas eletromecânicos que substitui. Ele possibilita a execução de sequências complexas de operação e cobre uma área muito maior de alternativas de aplicação. Além disso, permite que a temporização e sequência de operações sejam alteradas, bastando, para isso, mudar o software.

Uma aplicação bastante simples do micro como o "cérebro" de um sistema de controle refere-se ao alarme residencial contra ladrões. Este consiste, basicamente, em uma série de fios conectados a interruptores localizados em várias portas e janelas da casa. Quando todas estão fechadas, o circuito se completa e a corrente elétrica pode fluir pelo mesmo. Se o circuito for interrompido pela abertura de uma janela ou porta, o alarme soa.

Um dos problemas com sistemas desse tipo é a possibilidade relativamente alta de um falso alarme, provocado por falhas mecânicas ou mau contato nos interruptores. O computador oferece uma opção eficiente para a solução desse problema. Se, por exemplo, acrescentarmos um segundo circuito a todas as portas internas, e conectarmos ambos os circuitos a um microcomputador, o software poderá realizar alguns testes simples, antes de soar o alarme. Isso seria feito segundo uma determinada lógica: se um ladrão entrar na casa, uma porta interna deverá ser aberta logo após a interrupção do circuito externo. O alarme será ativado se os dois circuitos forem interrompidos na ordem correta — ou seja, do exterior para o interior. Caso os circuitos independentes sejam interrompidos isoladamente, ou em ordem incorreta, o alarme não é ativado.

O uso do computador no controle do sistema antiladrões apresenta outras vantagens. Por exemplo: dispondo de um sistema adequado de telecomunicações, podemos fazer com que o micro disque automaticamente um número de telefone e avise a polícia.

Em algumas residências, um interruptor simples é utilizado para ligar e desligar luzes internas, para dar a impressão — na ausência dos moradores — de que há gente em casa. Mas o feitiço pode se virar contra o feiticeiro, pois a sequência regular de liga-desliga evidencia exatamente o oposto: que não há

ninguém! Com um micro, é possível controlar tanto a iluminação dos aposentos como o funcionamento dos eletrodomésticos ligando-os segundo um padrão complexo, diferente para cada período do dia e variável de um dia para outro. Seqüências diversas podem ser deflagradas por eventos externos, como o nascer ou o pôr-do-sol.

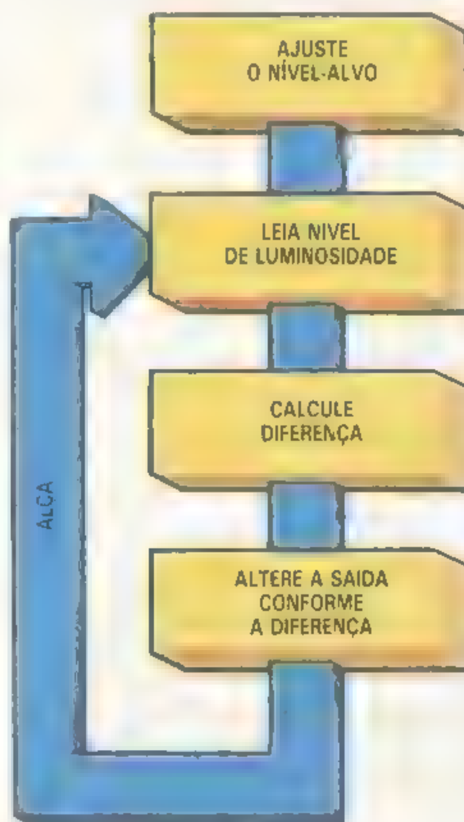
Outro candidato ideal para controle por micros é a ferrovia-modelo: um software adequado pode ser usado para verificar cruzamentos, acionar semáforos, desviar trilhos etc. Também interessante, no plano doméstico (ou até profissional), é o uso do micro para controlar um ou mais projetores de slides, de modo a criar um show mais complexo. O micro define a intensidade das lâmpadas — permitindo a criação de *fade-in* e *fade-out*, mixagens de imagens de dois projetores etc. —, a posição dos slides numa dada sequência e informa qual imagem será projetada em cada máquina. O micro encarrega-se, ainda, da sincronização de música e fala, gravadas em uma fita cassete.

CONTROLES DE REGULAÇÃO

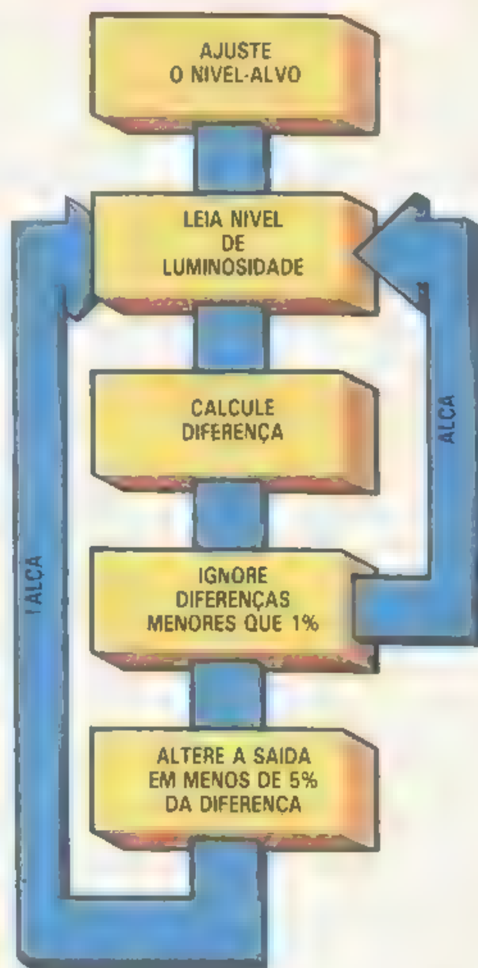
O controle de regulação é usado em toda aplicação em que a diferença entre uma situação ideal e a situação atual determina a operação a ser efetuada. Essa diferença é avaliada a partir de sensores conectados ao sistema. Um controlador de aquecimento central, por exemplo, tem a função de manter constante a temperatura do ambiente. Um sensor informa a temperatura do ambiente ao sistema de controle e este liga ou desliga o aquecedor conforme a temperatura caia ou suba.

Um microcomputador substitui facilmente sistemas desse tipo, permitindo operações mais complexas, como controle simultâneo de aposentos, do tanque de armazenamento de água quente etc. Para isso, ele aciona válvulas, bombas e elementos de aquecimento conforme a necessidade. Este sistema é um exemplo de um *servomecanismo* — baseado em controle por *alça fechada* (*feedback*, ou retroalimentação) — no qual a correção é proporcional ao erro detectado (*controle proporcional*).

Examinando mais detidamente o conceito de alça fechada, entenderemos por que o computador é tão versátil. Suponhamos que você queira regular a intensidade de uma lâmpada incandescente, de modo que o nível de luminosidade do ambiente seja constante. A variação da intensidade deve refletir as mudanças da luminosidade externa, ou seja, a lâmpada ficará mais fraca durante o dia, e mais forte à noite. Para isso, o computador é conectado a uma interface, que controla a corrente elétrica fornecida à lâmpada, e a um sensor fotelétrico, que mede o nível de luz no ambiente e fornece ao computador essa informação. Um sistema de retroalimentação de alça fechada é então incorporado, na forma de um software adequado, segundo este esquema:



Quando o programa for executado, a luminosidade da lâmpada começará a oscilar, tornando-se mais forte e, em seguida, mais fraca que o nível pretendido (nível-alvo). Esse efeito decorre de um atraso no sistema de controle, causado pela defasagem térmica da lâmpada — ou seja, ela não é capaz de responder à velocidade dos comandos do computador. Quando a corrente elétrica para a lâmpada é alterada, seu filamento demora a esquentar ou esfriar. Entretanto, o computador já determinou uma outra medida de luminosidade, e, comprovando que ainda existe um erro, ou diferença, aumenta o grau de correção. Eventualmente, a lâmpada "alcança" o computador e, se a correção efetuada é grande demais, chega a "ultrapassá-lo", invertendo o processo. Para evitar essa oscilação, algumas regras simples devem ser acrescentadas:



Agora, o sistema irá ajustar gradualmente a intensidade da luz e mantê-la constante, no nível-alvo. Outras regras podem ser incorporadas para a promoção de ações corretoras no caso de mudanças abruptas ou transitórias na iluminação externa.

REDUÇÃO DE DADOS

Aquisição e redução de dados e interfaces homem-máquina são elementos importantes de qualquer sistema de controle. Em muitas aplicações, o micro recebe informações de fontes diferentes, e tem que convertê-las para uma forma mais inteligível ou adequada. O software deve verificar se os níveis permanecem dentro de certos limites, converter unidades de medidas, filtrar os dados e checar inconsistências lógicas. Esse conjunto de atividades é chamado *redução de dados*.

Um analisador automotivo, para "acertar" a ignição de motores, constitui um bom exemplo de aplicação para as técnicas de redução de dados. Um sistema desse tipo exigirá sensores capazes

de monitorar o nível de dióxido de carbono na exaustão, o sincronismo da ignição, o ângulo de avanço do distribuidor etc. Ele poderia ser interativo — ou seja, o micro daria todas as instruções para a ligação dos sensores ao motor, acusaria eventuais falhas na operação e até mesmo diagnosticaria o problema do motor.

INTERFACES HOMEM-MÁQUINA

O que chamamos interface homem-máquina é exatamente o projeto da interação do sistema com o usuário. Esse elemento dos sistema de controle oferece enormes possibilidades de desenvolvimento. Mesmo ao nível mais simples, que consiste na exibição de determinada informação, há numerosas alternativas, como a simulação no vídeo de um mostrador analógico ou uma barra iluminada, em vez de complexas tabelas de números ou outros tipos de gráfico com os quais o usuário está menos habituado. As opções incluem a representação de um esquema do sistema que está sendo controlado. No caso de uma ferrovia de brinquedo, por exemplo, é possível traçar no vídeo os trilhos e os pontos de cruzamento, e assinalar em tempo real o seu "status". O usuário pode comandar todas as funções — como selecionar trens, mudar sua velocidade etc. — por meio de joysticks ou canetas ópticas.

O micro também abre novas perspectivas para o projeto e controle de sistemas de auxílio para deficientes físicos. Esses sistemas permitem, por exemplo, que um paralítico controle seu ambiente por meio de um computador, ligando ou desligando eletrodomésticos, alterando o nível da calefação ou a iluminação de um aposento etc.

Já existem diversas interfaces homem-máquina especiais para deficientes, como as de entrada e saída baseadas na voz ou aquelas em que se acionam as teclas soprando-se em um tubinho. Com isso, uma pessoa deficiente pode até mesmo usar um processador de textos, sem necessidade do teclado.

SENSORES

As formas de controle por meio de computadores que acabamos de examinar são bastante inter-relacionadas, e apresentam uma série de elementos em comum. Um dos mais importantes é a maneira como os computadores obtêm informações do mundo exterior.

Sensores especializados, correspondentes aos cinco sentidos — tato, olfato, visão, audição e paladar —, podem ser ligados a um microcomputador. Eles não chegam a ter o nível de complexidade dos sentidos humanos. Em compensação, os micros contam, potencialmente, com "sentidos" adicionais, como a capacidade de detectar campos magnéticos, radiação atômica, eletromagnética, de ultra-som ou raios X, infravermelha ou ultravioleta etc.

O computador precisa de um sensor específico para cada tipo de fenômeno a ser detectado. Existem, assim, detectores de som e ultra-som, temperatura, gases, fluxo, umidade, luz, proximidade, movimento, aceleração etc.

Seja qual for o tipo de energia à qual o sensor é sensível, geralmente a forma de saída é de natureza elétrica — como uma corrente ou uma voltagem analógica ou digital. Dá-se o nome de *transdutor* ao sensor que se encarrega de realizar a conversão de um tipo de energia para outro.

Como sabemos, o computador não pode trabalhar diretamente com sinais analógicos: é preciso, antes, convertê-los para sinais digitais, ou números binários. Isso é feito por uma interface especial de conversão, chamada *conversor analógico-digital* (AD).

Alguns sensores dispõem de algum tipo de circuito de pré-processamento, embutido — como amplificadores de sinal, condicionadores, filtros ou conversores AD simples — o que facilita o trabalho de conexão ao computador. Os detectores de temperatura (termistores), por exemplo, não respondem de forma inteiramente linear à variação da temperatura. Alguns termistores "inteligentes", porém, já apresentam o sinal de saída corrigido conforme o nível de temperatura, dispensando correções por parte do software de controle.

ATUADORES

Até agora, vimos como o computador "sente" o que está acontecendo no ambiente. Entretanto, para agir sobre o mesmo, ele precisa de atuadores, elementos que geram alguma forma de energia: luminosa, elétrica, mecânica, magnética etc. Na maioria das aplicações, os atuadores são eletromecânicos, dividindo-se, basicamente, em cinco tipos:

- pneumáticos
- hidráulicos
- motores elétricos DC ou AC
- solenóides
- motores de passo

Os atuadores pneumáticos, bem como os hidráulicos, utilizam a pressão gerada por fluidos (gases ou líquidos) e compõem-se de um cilindro contendo um pistão, conectado a um braço de impulsão ou de tração. Uma válvula operada eletricamnte dirige o fluxo de líquido ou gás para o pistão. Os atuadores hidráulicos geralmente são mais seguros, potentes e precisos do que os pneumáticos.

Motores comuns de corrente direta (DC) ou alternada (AC) podem ser controlados por um micro, embora a precisão seja pequena. Quando é necessário um posicionamento exato, deve-se utilizar um sensor do ângulo de rotação do eixo, que informa ao computador a posição e velocidade. Esses codificadores, como são chamados, transformam o ângulo de rotação em um número de bits. Podem ser ópticos ou elétricos — nesse último caso, temos um conjunto *servomotor*.

Devido às complicações mencionadas e ao alto preço de um servomotor, os sistemas baseados em micros usam com maior freqüência os solenóides ou os motores de passo. Mais simples — já que precisam do computador só para ligá-los e desligá-los, não exigindo controle proporcional — esses atuadores adaptam-se melhor ao mundo digital.

Um solenóide é um eletroímã que aciona algum tipo de eixo (ou armadura). Quando a corrente elétrica passa através de uma bobina, o campo magnético resultante move a armadura. O relé é um solenóide, só que destinado a ligar e desligar correntes elétricas.

O motor de passo (*stepper*) utiliza uma armadura circular de solenóides para girar um eixo, do mesmo modo que um motor elétrico comum. A diferença é que o avanço se faz em pequenos passos, e não em movimentos contínuos. A utilização do stepper em conjunto com um micro é simples, mas esse atuador apresenta uma desvantagem: sua potência é bem inferior à de motores elétricos de tamanho semelhante.

Para entender o funcionamento de um motor de passo, tomemos como exemplo um modelo simplificado com apenas quatro pólos, ou seja, quatro eletroímãs dispostos em ângulos retos ao redor do rotor. O rotor é uma armadura de ferro, com duas peças polares, e gira livremente sobre um eixo. Se um dos magnetos é acionado, o rotor é atraído para ele, e gira um certo número de graus. Caso um microcomputador acione em seqüência cada um dos magnetos, o rotor girará um número preciso de vezes — efeito impossível de se obter com um motor linear. Quanto mais

rápida a atuação do micro, mais depressa irá girar o rotor. Os motores de passo têm um grande número de bobinas (cerca de duzentas, em média), e, consequentemente conseguem uma rotação muito mais "macia". O rotor pode ser posicionado sem necessidade de servomecanismos que indiquem sua posição.

CONEXÃO AO COMPUTADOR

A maioria dos sensores e atuadores exige ou fornece uma voltagem entre cinco e dez volts, muitas vezes com uma corrente de valor elevado. Como o micro não foi projetado para trabalhar diretamente com essas voltagens e correntes, a conexão às portas de entrada e saída exige uma interface.

As interfaces disponíveis oferecem saídas comutadas de corrente ou voltagem de valor elevado, entradas por interruptores, portas binárias de oito a dezesseis bits, conversores AD ou DA de média ou alta velocidade etc.

Muitos micros possuem conversores AD embutidos (a porta de gravador cassete, por exemplo, ou a entrada de joysticks), mas estes não têm frequências de amostragem (número de conversões feitas por segundo) tão altas quanto as requeridas por muitas aplicações. Por isso, conversores extras são necessários. Um conversor DA, por sua vez, fornece voltagens ou correntes contínuas, a partir de sinais binários, para acionar motores DC, lâmpadas, alto-falantes e assim por diante.

Alguns sistemas de controle doméstico, como os destinados a ligar e desligar eletrodomésticos a distância, utilizam a própria rede de força da casa para enviar os sinais binários.

SISTEMAS MECÂNICOS

Existem várias alterativas para a ligação do computador a um atuador mecânico. Muitas delas foram usadas inclusive no desenvolvimento de brinquedos ou kits acionados por computadores. No exterior, encontram-se diversos sistemas — como o Meccano, o Lego e o Fischer — que são verdadeiros projetos de engenharia em miniatura, com todos os componentes imagináveis.

Quanto maior a flexibilidade do kit, melhor, pois conexões mecânicas quase sempre precisam ser projetadas de acordo com a tarefa. Um sistema de engrenagens, polias e alavancas, conectadas a relés, solenóides e motores, requer, é evidente, um minucioso trabalho de planejamento e construção.



SOFTWARE

Desenvolver software para aplicações de controle não é mais difícil do que para outros tipos de aplicação, como jogos, por exemplo. As interfaces disponíveis simplificam bastante esse trabalho. Muitas vezes, elas são fornecidas com uma biblioteca de rotinas em linguagem de máquina ou BASIC, que podem ser utilizadas como módulos de um sistema mais complexo de controle, escrito pelo usuário.

Um dos passos mais importantes no desenvolvimento do software consiste na análise e divisão do problema de controle em partes menores, de tal forma que se possa escrever e testar individualmente cada seção, antes de integrá-la ao sistema maior.

A necessidade de manter o microcomputador permanentemente conectado ao sistema de controle costuma desestimular o usuário a se aventurar nessa área. Esse inconveniente, porém, pode ser superado com a aquisição de um *sistema de desenvolvimento*, que é um micro mais simples, geralmente em uma única placa, específica para as tarefas de controle. Essa solução, além de ser barata, é a mais adequada: dificilmente uma aplicação de controle requer todos os recursos disponíveis em um computador de uso geral.

Os sistemas de desenvolvimento são fornecidos com um conjunto de hardware e software que permite a adaptação do computador à tarefa específica que será realizada. Se os dados a serem exibidos são exclusivamente numéricos, basta um visor LED ou de cristal líquido, do tipo existente para calculadoras. Caso a aplicação exija apenas algumas teclas ou botões, não é preciso acrescentar um teclado completo. Entretanto, sistemas desse tipo têm necessidades que um micro comum nem sempre é capaz de atender — como um número grande de canais analógicos, ou saída de controle de correntes altas.

O usuário pode modificar à vontade o software que acompanha os sistemas de desenvolvimento, incorporando-o, posteriormente, a uma memória EPROM, para uso permanente.

O LOGO E A TARTARUGA

Estrelas, círculos, espirais: como [illegible] figuras surgem no vídeo — por obra [illegible]

[illegible] comandos gráficos do LOGO usando a tartaruga, uma espécie de cursor triangular que indica a posição atual na tela. Podemos também "ensinar" a tartaruga a traçar determinadas formas através de um comando especial. Se quisermos, por exemplo, fazê-la desenhar [illegible]

```
[illegible]
REPEAT 6 [illegible]
[illegible]
```

```
AP HEXAGONO
REPITA 6 [PARAFRENTE [illegible]
PARADIREITA 60]
FIM
```

[illegible] END, o interpretador indicará que o procedimento HEXAGONO foi definido, passando a fazer parte do vocabulário LOGO.

Quando você digitar a palavra HEXAGONO, a tartaruga desenhará um polígono regular de seis lados (cada lado com cinqüenta passos de comprimento), a partir da posição em que estiver. Depois, ela voltará ao ponto inicial, por percorrer seis vezes 60 graus, ou seja, 360 graus. Esse "comportamento" [illegible] se repete com polígonos regulares, [illegible] recebido um nome jocoso dos [illegible] do LOGO: Teorema do [illegible] Total da Tartaruga (TTTT).

Um procedimento pode ser utilizado dentro de outros, como mostra o programa abaixo, chamado FLOR, que desenha doze hexágonos ao redor de um [illegible]

```
[illegible] FLOR
REPEAT 12 [illegible]
30]
[illegible]
```

```
AP FLOR
REPITA 12 [HEXAGONO PF 10 PD
30]
FIM
```

[illegible] passos adiante, antes de [illegible] o traçado do hexágono [illegible] sua direção para [illegible] da direção anterior.

A VISAO DA TARTARUGA

Para fazer desenhos no vídeo, normalmente o BASIC usa como referencial as coordenadas de um sistema cartesiano (de eixos ortogonais X e Y), com o qual estamos mais familiarizados. A geometria da tartaruga é diferente: nela, o ponto de referência é a própria tartaruga, ou seja, todos os deslocamentos lineares e angulares são realizados a partir da última posição [illegible] tartaruga se encontrava.

É por isso que [illegible] desenhar um hexágono, e não 120 graus, que corresponde ao ângulo entre [illegible] lados [illegible] ou seja, temos que [illegible] a tartaruga 60 graus para produzir um ângulo de 120 graus [illegible] traçada [illegible] próxima.

Parece confuso? [illegible] mais fácil aprender geometria de uma forma intuitiva como esta, que nos permite "experimentar" o referencial da tartaruga, bastando imaginar que estamos em uma sala vazia, traçando uma figura com nossos passos.

PROCEDIMENTOS COM VARIAVEIS

Embora o procedimento HEXAGONO tenha sido incorporado ao vocabulário do LOGO, há uma importante diferença entre ele e vários dos comandos primitivos da linguagem, como aqueles que fazem a tartaruga avançar, recuar ou virar: o HEXAGONO traça uma figura sempre do mesmo tamanho, enquanto que os comandos mencionados admitem um parâmetro variável.

Entretanto, é perfeitamente possível definir procedimentos que tenham parâmetros variáveis. Para isso, precisamos atribuir um nome ao parâmetro [illegible] e incluí-lo na linha de título [illegible]

[illegible] dido de dois pontos, que indicam [illegible] interpretador que se trata de uma variável. Para usar a variável dentro do procedimento [illegible]



[illegible] pontos antes.

Se você quiser definir um procedimento **HEXAGONO** que lhe permita especificar a medida do lado no momento de desenhar, faça: [illegible]

```
TO HEXAGONO :LADO
REPEAT 6 [FORWARD :LADO RIGHT
60]
END
```

```
AP HEXAGONO :LADO
REPITA 6 [PARAFRENTE :LADO
PARADIREITA 60]
FIM
```

Agora, [illegible] bastará digitar a palavra do procedimento, seguida do número correspondente ao comprimento desejado para o lado. Se você se esquecer de colocar o número, o interpretador LOGO [illegible] uma mensagem de erro.

Digitando **HEXAGONO 20**, por exemplo, você fará a tartaruga desenhar um hexágono com vinte unidades (ou passos) de lado; com **HEXAGONO 40** obterá um outro hexágono com quarenta unidades de lado. O uso de :LADO dentro da linha de repetição do procedimento provocará a substituição automática do valor numérico anteriormente especificado na entrada do comando, **HEXAGONO**, [illegible] funcionar como os primitivos do LOGO.

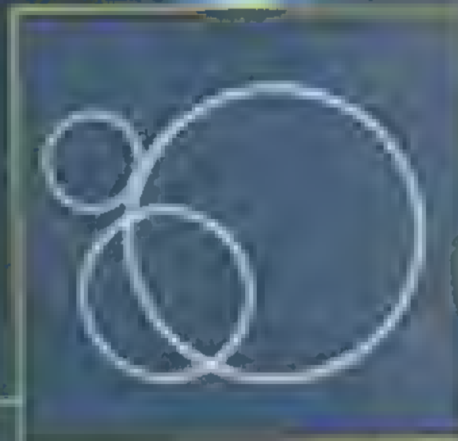
Façamos o mesmo com **FLOR**, [illegible] o procedimento: [illegible]

```
TO FLOR :LADO
REPEAT 12 [HEXAGONO :LADO RT 30]
END
```

```
AP FLOR :LADO
REPITA 12 [HEXAGONO :LADO PD 30]
FIM
```

Quando [illegible] **FLOR**, deveremos fornecer o tamanho do lado. Experimente várias medidas para ver o resultado. Do mesmo modo, podemos tornar variáveis o avanço [illegible] [illegible]







```
REPEAT_ :NUMEROLADOS [FORWARD
:LADO RIGHT 360/:NUMEROLADOS]
```

POLIGONO



POLIGONO



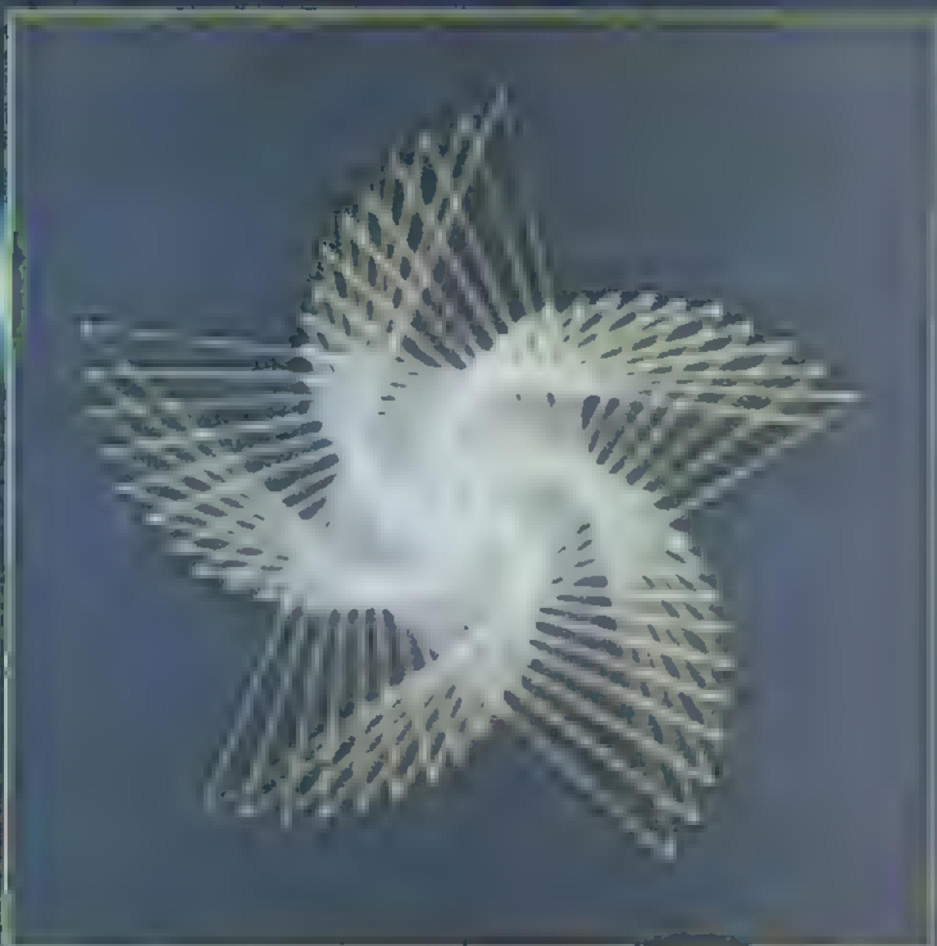
parecer a tartaruga, digitamos:

to que adquirimos até agora:

```
HIDETURTLE
REPEAT 72
END
```

```
AP CIRCULO
```

MODIFICACAO E ARMAZENAGEM



cução ou modificação.

O LOGO dispõe de um editor sim-

EDIT (EDIT)

procedimento a ser editado.

O comando EDIT apaga a tela e lista

CTRL BREAK





ORGANIZAÇÃO DA MEMORIA

REPETICAO E RECURSAO

o FLOR, utilizando a recursão:

```
END
```

```
AP ESPIRAL :LADO :ANGULO
ESPIRAL :LADO+2 :ANGULO
```

temos uma bonita espiral.
Tente os seguintes valores:



ça de tornar a tartaruga invisível.

COMO PARAR

MICRO DICAS

TRADUÇÃO PARA O MLOGO

Os comandos do MLOGO, para o Apple, correspondentes aos comandos do BRASLOGO usados neste artigo, são:

BRASLOGO	MLOGO
SEMTARTARUGA	SEMT
EDITE	EDITE
SE	SE
ENTAO	ENTAO
PARE	PARE
GRAVETUDO	GRAVE
CARREGUE	LEIA
MOPS	LISTAR
MO	LISTE
ELIMINE	
ELIMINEARQ	SEMARQUIVO
ARQUIVOS	VERDISCO

```
SE :LADO>60 ENTÃO [PARE]
```

tada se a condição SE for verdadeira

recursão, no programa GIRAHEX?

COMPRESSÃO DE TEXTOS (1)

As limitações da memória do micro costumam frustrar os programadores que apreciam jogos de aventura. Veja como fazer para colocar o máximo de texto em um mínimo de memória.

Todo programador enfrenta o mesmo problema ao desenvolver jogos de aventura: o texto não cabe no espaço disponível na memória do micro. Quem não pode comprar uma maquina maior ou expandir a memória da sua, costuma contornar essa dificuldade reduzindo o tamanho do jogo ou simplificando demais o texto. Muitas vezes, o programador se vê obrigado, por exemplo, a eliminar as instruções do programa — o que é mau para quem vai utilizá-lo.

Existe uma maneira, porém, de tornar o programa menos extenso. Os métodos normalmente empregados para encurtar um programa, discutidos no artigo da página 141, não são eficientes no que diz respeito a jogos de aventura, pois a maior parte destes consiste em textos (listas de objetos, locações, situações, mensagens, instruções etc.). A solução está, portanto, em reduzir o espaço ocupado pelo texto.

Examinaremos aqui várias técnicas, de compressão de texto e, em artigo posterior, veremos como usar uma delas em um jogo de aventuras.

COMO UM TEXTO É ARMAZENADO

O computador normalmente armazena o texto — ou seja, tudo o que não é instrução de programa, ou constante numérica — da mesma maneira em que foi entrado, usando o código ASCII (veja o artigo da página 361).

Recordando, o código ASCII (*American Standard Code for Information Interchange*, Código Padrão Americano para Intercâmbio de Informação) é um padrão utilizado para representar caracteres na memória do computador. Ele comporta 256 códigos ou caracteres diferentes. Os códigos 0 a 31 são reservados para funções de controle e os códigos 32 a 90, para caracteres alfabéticos, numéricos e de pontuação. Os códigos 91 e seguintes estão livres para outros usos, como caracteres gráficos, sinais de acentuação.

Duas características do ASCII são especialmente relevantes para o problema de compressão de textos:
- um código ASCII ocupa um byte de memória (ou oito bits), pois 255 é o maior número que pode ser armazenado em oito bits (11111111, em binário);
- o código ASCII possui mais códigos numéricos do que caracteres para representar (e, por isso, é chamado *degenerado*). Poderia, portanto, ser diminuído, sem muito sacrifício.

Na realidade, o ASCII é uma evolução de sistemas de códigos anteriormente existentes, que ocupavam menor espaço de memória, pois tinham sete ou seis bits (como o código Baudot, usado em máquinas de telex). A ampliação do espaço ocupado para oito bits significou uma adaptação ao mundo dos computadores digitais, onde predominam os processadores com número de bits organizados em potências de 2 — quatro, oito dezesseis, 32 bits etc. Com isso, expandiu-se no ASCII o alcance original dos códigos de texto.

Se os códigos dos caracteres fossem comprimidos em menos de oito bits, o espaço total para a armazenagem de texto na memória seria reduzido proporcionalmente. Podemos fazer isso adotando em nosso microcomputador um código diferente do ASCII e escrevendo um programa que faça a tradução do ASCII para o nosso código particular e vice-versa. Essa tradução é necessária porque os periféricos de entrada e saída, assim como as funções de programação que trabalham com textos, obedecem ao padrão ASCII. Existem diversas alternativas para a elaboração de um programa desse tipo, cada uma com suas vantagens e desvantagens.

DE OITO PARA SEIS

A maneira mais direta e intuitiva de se comprimir um texto consiste em utilizar apenas uma parte do código ASCII. De modo geral, qualquer texto pode ser escrito com o emprego de apenas 128 códigos, que cabem em sete bits. A economia de espaço obtida, nesse caso, é da ordem de 1 em 8, ou, em termos percentuais, de 12,5%.



É possível também escrever um texto empregando caracteres maiúsculos, numerais e sinais de pontuação, que correspondem aos códigos ASCII 32 a 90, ou seja, a 59 códigos. Se usarmos seis bits (o que permite armazenar um máximo de 64 códigos), conseguiremos uma redução de 2 em 8, ou de 25%, o que é bem razoável para muitas aplicações.

Converter o código ASCII ao nosso novo código de seis bits é fácil: basta diminuir 32 do código ASCII. O resultado será um código com valores de 0 a 58, com os caracteres representados exatamente na mesma ordem. Para a conversão no sentido inverso, basta somar 32 — e teremos o código ASCII, de novo.

- ARMAZENAGEM DO TEXTO
- ECONOMIA DE ESPAÇO
- TEORIA DA COMPRESSÃO
- CODIGO ASCII
- O USO DA ESTATISTICA

A redução pode ser ainda maior. Um código de cinco bits permite que se faça a representação de 32 caracteres: todas as letras maiusculas, que são 26, e mais seis caracteres destinados a pontuação — por exemplo, o espaço em branco, o ponto, a vírgula, os dois pontos, o hífen e o sinal de interrogação. Se o texto contiver números, eles terão que ser escritos por extenso, o que, na verdade, não representa um problema muito grande.

Com um código de cinco bits, conseguimos uma redução de 5 em 8, ou seja, de 37,5%, o que significaria uma economia de quase quatro Kbytes em cada dez Kbytes de texto. Uma redução, convenhamos, considerável.

UM PROGRAMA DE CONVERSÃO

A conversão do ASCII e para o ASCII, nesse caso, não é tão fácil assim, exigindo um programa específico, como o que se segue:



```
10 DIM F$(50)
20 K$="ABCDEFGHIJKLMNOPQRSTUVW
XYZ ,.:-?"
100 CLS
180 PRINT "CODIFICANDO..."
190 I=0:NC=0
200 READ L$:IF L$="*" THEN 230
210 GOSUB 710:I=I+1:F$(I)=X$
215 NC=NC+LEN(X$)
220 GOTO 200
230 STOP
700 REM --- CODIFICACAO
710 X$="":FOR J=1 TO LEN(L$)
720 C$=MID$(L$,J,1)
730 P=INSTR(K$,C$):IF P>0 THEN
X$=X$+CHR$(P)
740 NEXT J:RETURN
```

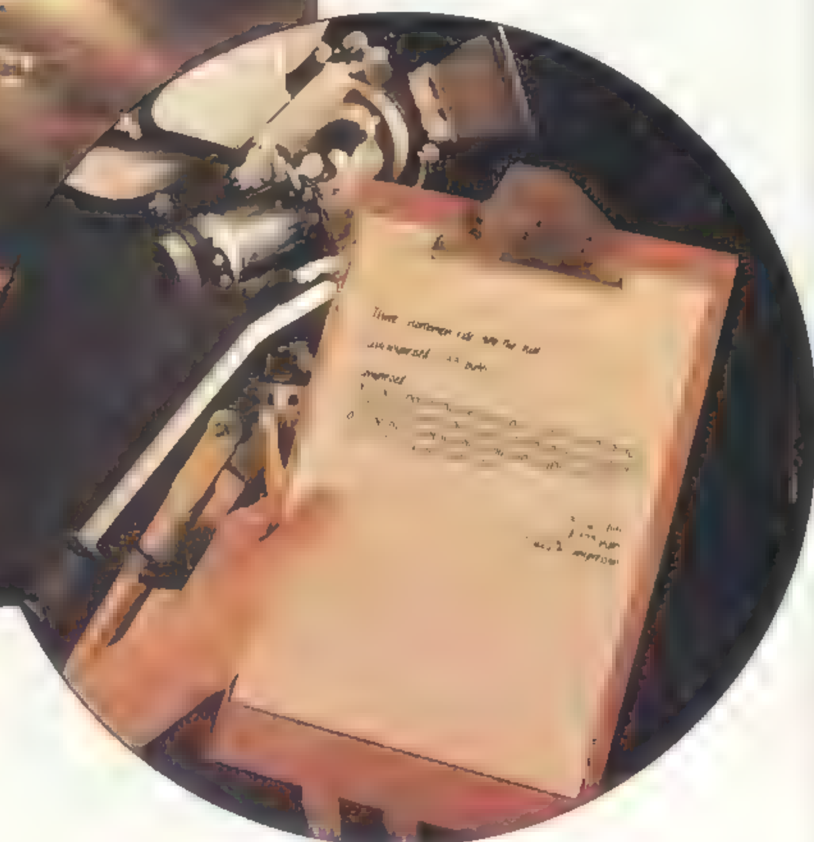
Coloque o comando **CLEAR 1000** na linha 10, para micros das linhas TRS-80 e TRS-Color.



```
10 DIM F$(50)
20 K$="ABCDEFGHIJKLMNOPQRSTUVW
   XYZ .,:-?"
100 HOME
180 PRINT "CODIFICANDO..."
190 LET I=0:NC=0
200 READ L$:IF L$="*" THEN 230
210 GOSUB 710:I=I+1:F$(I)=X$
215 LET NC=NC+LEN(X$)
220 GOTO 200
230 STOP
700 REM --- CODIFICACAO
710 LET X$="":FOR J=1 TO
    LEN(L$)
720 LET C$=MID$(L$,J,1)
725 FOR P=1 TO 32
730 IF MID$(K$,P,1)=C$ THEN X$
    = X$ + CHR$(P)
735 NEXT P
740 NEXT J:RETURN
```



```
10 DIM f$(50,64),k$(32)
20 LET k$="ABCDEFGHIJKLMNOPQRST
UVWXYZ .,:-?"
180 PRINT "CODIFICANDO..."
190 RESTORE:LET i=0:LET nc=0
200 READ L$:IF L$="*" THEN GOTO
230
210 GOSUB 710:LET i=i+1:LET
f$(i)=x$
215 LET nc=nc+LEN x$
220 GOTO 200
230 STOP
700 REM --- CODIFICACAO
710 LET x$="":FOR j=1 TO LEN L$
```



```
720 LET c$=L$(j TO j)
725 FOR p=1 TO 32
730 IF k$(p TO p)=C$ THEN X$=X$
+ CHR$ P
735 NEXT p
740 NEXT j:RETURN
```

Para testar este e outros programas de codificação de textos, podemos colocar no fim do programa várias linhas **DATA**, contendo, por exemplo, as instruções de um jogo de aventura (a última linha deve ter um asterisco):

```
5000 DATA "POR MAIS DE DOIS MIL
 ANOS, O AMULETO SAGRADO DE NIT
PU FOI"
5010 DATA "ZELOSAMENTE GUARDADO
 PELOS ALDEAES DO CONDADO DE NO
RDHAM."
5020 DATA "A SUA POSSESSAO ERA
A CHAVE PARA A SEGURANCA E A FE
LICIDADE"
5030 DATA "DO REINO DE DUCHESS,
 NA ESCOCIA. O CONDE, QUE ERA O
 PROPRIETARIO"
5040 DATA "DE TODO AS TERRAS AO
 REDOR DA ALDEIA, JUROU DEFENDE
R O AMULETO"
5050 DATA "COM SUA PROPRIA VIDA
"
5060 DATA "UM DIA, HORROR E DES
GRACA SE ABATEM SOBRE A ALDEIA.
 O"
5070 DATA "AMULETO SAGRADO TINH
A DESAPARECIDO ! UMA MENSAGEM E
SCRITA"
5080 DATA "COM SANGUE NA PAREDE
 DO SANTUARIO PROCLAMAVA AOS QU
ATRO"
5090 DATA "VENTOS O NOME DO FAC
INORA: O CRUEL REI DE AARDVARK,
 SOBERANO"
5100 DATA "NORMANDO QUE HABITAV
A UM CASTELO SOMBRIO E CHEIO DE
 ARMADILHAS"
5110 DATA "E DE ONDE SO' SE OUV
EM OS GRITOS DOS PRISIONEIROS T
ORTURADOS."
5120 DATA "VOCE E' O GALANTE CO
NDE DE NORDHAM, E SUA MISSAO E'
 RECUPERAR"
5130 DATA "O AMULETO SAGRADO. U
M PLANO DESESPERADO FORMA-SE EM
 SUA MENTE:"
5140 DATA "A UNICA MANEIRA DE E
NTRAR NO CASTELO E' DEIXAR-SE A
PRISIONAR"
5150 DATA "PELOS ASSECLAS FEROZ
ES DE AARDVARK. PARA COMPLETAR
A AVENTURA"
5160 DATA "VOCE PRECISA SE DESV
ENCILHAR DE SEUS CAPTORES, PERC
ORRER OS"
5170 DATA "LABIRINTOS MORTAIS D
O CASTELO, ACHAR O AMULETO, E D
EPOIS"
5180 DATA "ESCAPAR."
5190 DATA "SEUS RECURSOS SAO PO
UCOS: UMA ESPADA, UMA TOCHA, UM
 PUNHADO"
5200 DATA "DE ALIMENTOS SECOS,
```

```
UM CANTIL D'AGUA, E UM PUNHAL.
A SUA FRENTE."
5210 DATA "UMA SEQUENCIA DE BAN
DIDOS E FERAS DE ARREPIAR OS CA
BELOS..."
5220 DATA "BOA SORTE !"
5230 DATA "*"
```

Se preferir, digite apenas as primeiras linhas. Só precisaremos deste texto completo quando formos testar programas mais complexos, adiante.

COMO FUNCIONA

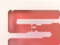
O programa de codificação, que transforma o código ASCII em um código reduzido de cinco bits, está contido no laço das linhas 200 a 220. A linha 200 lê uma linha de texto, **L$**, em **DATA**, terminando ao encontrar um asterisco. A linha 710, por sua vez, chama a sub-rotina de codificação, que verifica se cada caractere **C$**, extraído de **L$**, está na lista de codificação armazenada em **K$** pela linha 20 do programa principal. Se estiver, sua posição sequencial **P**, em **K$**, é usada como código, e um **CHR$(P)** concatena este caractere em uma cadeia de saída, **X$**. Se não estiver, então nada é concatenado.

O conteúdo de **X$** é armazenado na lista **F$**, onde cada elemento corresponde a uma linha do texto em **DATA**. A armazenagem é necessária para uso posterior. A variável **NC** da linha 225 serve simplesmente para contar o número de caracteres convertidos.

Como os textos armazenados em **F$** já não se encontram mais em ASCII, não podemos listá-los com um **PRINT** normal, pois muitos deles correspondem a caracteres ASCII de controle e provocariam a maior desordem na tela

Se quiser ver como ficam os códigos numéricos convertidos, acrescente estas linhas ao programa e rode-o:

```
212 GOSUB 820
800 REM --- LISTAGEM
820 FOR N=1 TO LEN(X$)
830 PRINT ASC(MID$(X$,N,1));
840 NEXT N:PRINT
850 RETURN
```



```
212 GOSUB 820
800 REM --- LISTAGEM
820 FOR n=1 TO LEN x$
830 PRINT ASC X$(n TO n);
840 NEXT n:PRINT
850 RETURN
```

Para decodificar os textos — ou para reconvertê-los ao código ASCII —, devemos adicionar as linhas seguintes e rodar novamente o programa:



```
230 NL=I:T=0
235 PRINT "TOTAL: ";NC; "CARACT
ERES":PRINT"DECODIFICANDO..."
240 FOR I=1 TO NL
250 L$=F$(I):GOSUB 870
260 T=T+1:IF T<10 THEN 290
270 PRINT:INPUT"PRESSIONE <ENTE
R>";X$
280 T=0
290 NEXT I
300 STOP
860 REM --- DECODIFICACAO
870 FOR J=1 TO LEN(L$)
880 C=ASC(MID$(L$,J,1))
920 PRINT MID$(K$,C,1);
940 NEXT J
950 PRINT:RETURN
```



```
230 LET nl=I:LET t=0
235 PRINT "TOTAL: ";nc; "CARACT
ERES":PRINT"DECODIFICANDO..."
240 FOR i=1 TO nl
250 LET L$=f$(i)   GOSUB 870
260 LET t=t+1.IF t<10 THEN GOTO
290
270 PRINT:INPUT "PRESSIONE <ENT
ER>",x$
280 LET t=0
290 NEXT i
300 STOP
860 REM --- DECODIFICACAO
870 FOR j=1 TO LEN L$
880 LET c=ASC L$(j TO j)
920 PRINT K$(c TO c);
940 NEXT j
950 PRINT:RETURN
```

A rotina de decodificação (da linha 870 à 950) extrai cada caractere do texto codificado e troca seu código pelo caractere encontrado na mesma posição, na chave de codificação **K$**.

MANIPULAÇÃO DE BITS

Ainda não ganhamos espaço na memória: apesar de termos reduzido o conjunto de caracteres usados, eles continuam sendo armazenados em um byte (desperdiçando três bits em cada byte).

Para comprimir o texto de fato, precisamos utilizar os bits que estão sobrando, no seguinte esquema:

1º byte - 5 bits do 1º caractere
            3 bits do 2º caractere
2º byte - 2 bits do 2º caractere
            5 bits do 3º caractere
            1 bit do 4º caractere
3º byte - 4 bits do 4º caractere
            4 bits do 5º caractere

e assim por diante.

Esta tarefa de manipulação de bits pode ser cumprida sem grande dificuldade em linguagem BASIC — por meio dos operadores lógicos **AND, OR** e **NOT** —, porém, é lenta demais. Na verdade, o ideal seria realizá-la através de uma sub-rotina **USR** em linguagem de máquina, mas não vamos fazê-lo aqui, para não tornar muito complicada a programação do jogo de aventura

Optamos por um outro método de compressão, também baseado em manipulação de bits, mas que pode ser facilmente executado em BASIC. Esse método é bem mais eficaz que a restrição do conjunto de caracteres, pois garante uma redução do texto de cerca de 45%.

A programação, aqui, é mais simples, pois os bits não são manipulados individualmente, mas em grupos de quatro (meio byte ou *nibble*, em inglês). Com uma única expressão aritmético-lógica simples podemos ter acesso ou escrever nos nibbles individuais.

UM POUCO DE ESTATÍSTICA

Como sabemos, cada idioma utiliza determinadas letras com maior frequência do que outras. A técnica que escolhemos fundamenta-se exatamente na frequência do emprego das letras. Na língua portuguesa, por exemplo, catorze letras — A, S, E, D, R, O, I, U, N, C, M, T, P e L — representam cerca de 90% do total de um texto, juntamente com o espaço em branco.

Usaremos um código baseado apenas em quatro bits (15 é o maior número decimal que se pode representar com quatro bits, ou um nibble). Com isso, obteremos uma compressão de 50% do texto, o que é, teoricamente, o máximo que se pode conseguir. E será fácil fazer o programa, porque caberão sempre dois caracteres por byte.

Mas o que fazer com as outras letras? Mesmo sendo usadas com uma frequência muito baixa, elas são essenciais para compressão do texto. Precisam, por isso, estar presentes.

A solução também é espantosamente simples: podemos usar essas letras com seu código ASCII normal, de oito bits. No processo de codificação, toda vez que o programa encontrar uma letra não incluída no grupo das mais frequentes, ele atribuirá um código 0 a ela. Dessa maneira, indica-se ao programa decodificador que o que vem a seguir é um código ASCII de oito bits.

Uma letra incomum usa, assim, três nibbles (oito do código, mais quatro da baliza 0), ou um byte e meio: este é o preço que se paga por codificar a maio-

na das letras com quatro bits. A eficiência da compressão será menor, mas não muito. Façamos as contas, tomando um conjunto de cem caracteres:

90 têm códigos de 4 bits = 45 bytes
10 têm códigos de 12 bits = 15 bytes

TOTAL = 60 bytes

Conseguimos, teoricamente, uma compressão de 40%. Na realidade, ela será um pouco menor, dependendo do texto. Ainda assim, o resultado é bem melhor do que o obtido por uma compressão com códigos de seis bits.

Para verificar com que freqüência são empregadas as letras presentes em nosso texto-teste, digite o programa que se segue, acrescente as linhas **DATA** ao final e rode-o:



```
15 DIM F(60),C(60)
30 NT=0
40 FOR I=1 TO 60
50 F(I)=0:C(I)=I
60 NEXT I
65 PRINT "ANALISANDO..."
70 READ L$:IF L$="*" THEN GOTO
85
80 GOSUB 410:GOTO 70
85 PRINT "ORDENANDO..."
90 GOSUB 470:GOSUB 560
100 STOP
400 REM --- CONTAGEM
410 FOR I=1 TO LEN(L$)
420 NT=NT+1
430 J=ASC(MID$(L$,I,1))-31
440 F(J)=F(J)+1:NEXT I
450 RETURN
460 REM --- ORDENACAO
470 N=60
480 FL=0
490 N=N-1:FOR I=1 TO N
500 IF F(C(I))=>F(C(I+1)) THEN
520
510 X=C(I):C(I)=C(I+1):C(I+1)=X
:FL=1
520 NEXT I
530 IF FL=0 OR N=2 THEN RETURN
540 GOTO 480
550 REM --- IMPRESSAO
560 PRINT
570 PRINT "FREQUENCIA SIMPLES
DOS CARACTERES NO TEXTO":PRINT
580 FOR J=1 TO 60 STEP 3
590 FOR I=J TO J+2
600 IF F(C(I))=0 THEN GOTO 640
610 PRINT CHR$(C(I)+31);" ";
F(C(I)),
620 NEXT I:PRINT
630 NEXT J
640 PRINT:PRINT"TOTAL: ";NT;"
CARACTERES."
650 RETURN
```

Coloque um comando **CLEAR 3000** na linha 15, para os micros pertencentes às linhas TRS-80 e TRS-Color.

```
15 DIM f(60),c(60)
30 LET nt=0
40 FOR i=1 TO 60
50 LET f(i)=0:LET c(i)=i
60 NEXT i
65 PRINT "ANALISANDO..."
70 READ L$:IF L$="*" THEN GOTO
85
80 GOSUB 410:GOTO 70
85 PRINT "ORDENANDO..."
90 GOSUB 470:GOSUB 560
100 STOP
400 REM --- CONTAGEM
410 FOR i=1 TO LEN L$
420 LET nt=nt+1
430 LET j=ASC(L$(i to i))-31
440 LET f(j)=f(j)+1:NEXT i
450 RETURN
460 REM --- ORDENACAO
470 LET n=60
480 LET fl=0
490 LET n=n-1:FOR i=1 TO n
500 IF f(c(i))=>f(c(i+1)) THEN
GOTO 520
510 LET x=c(i):LET c(i)=c(i+1)
: c(i+1)=x:LET fl=1
520 NEXT i
530 IF fl=0 OR n=2 THEN RETURN
540 GOTO 480
550 REM      IMPRESSAO
560 PRINT
570 PRINT "FREQUENCIA SIMPLES
DOS CARACTERES NO TEXTO":PRINT
580 FOR j=1 TO 60 STEP 3
590 FOR i=j TO j+2
600 IF f(c(i))=0 THEN GOTO 640
610 PRINT CHR$(c(i)+31);" ";
f(c(i)),
620 NEXT i:PRINT
630 NEXT j
640 PRINT:PRINT"TOTAL: ";nt;"
CARACTERES."
650 RETURN
```

As linhas 15 a 60 definem e inicializam dois conjuntos: **F**, que conterá a freqüência de cada caractere presente no texto, e **C**, um conjunto-índice que será utilizado pela rotina de ordenação (no começo **C** contém os números inteiros de 1 a 60, em ordem crescente). A variável **NT** servirá para contar o número total de caracteres no texto.

O laço que abrange as linhas 70 e 80 lê as linhas de texto original em **DATA**, parando se encontrar um asterisco. Caso contrário, chama a rotina 410, que tem por objetivo executar a contagem. Essa rotina simplesmente acha o código ASCII de cada um dos caracteres presentes na linha de texto, diminui o valor 31 dos mesmos, e incrementa o elemento de **F** correspondente.

Em seguida, as sub-rotinas 470 e 560 (chamadas na linha 90 do programa) encarregam-se, respectivamente, de colocar o conjunto **C** em ordem decrescente de freqüência e mostrar o resultado na tela. A rotina de ordenação usa o método tipo bolha, que examinamos no artigo da página 468. A rotina de exibição mostra apenas os caracteres com freqüência maior do que 0.

Se tudo tiver corrido bem, você obterá uma tabela com estes dados:

```
   192      A  147      E  129
O  111      S  87       R  84
D  66       I  47       U  45
N  43       C  40       M  39
T  36       P  30       L  29
.  15       H  14          12
V  12       G  11       B  9
F  8           5        Ç  4
:  3           2        -  2
K  2        Z  2        J  1
X  1

TOTAL:  1228  CARACTERES
```

Note que, como afirmamos anteriormente, um conjunto de apenas catorze letras mais o espaço em branco respondem por 91,6% do texto (1125 ocorrências em 1228 letras). O espaço em branco é o caractere mais freqüente em qualquer texto, por razões óbvias.

Um segundo conjunto, formado por mais catorze caracteres, responde por 8% do texto. Caberia a um terceiro conjunto abrigar os demais caracteres (a maioria dos quais não apareceu nenhuma vez neste texto, mas poderia constar esporadicamente de outros).

Vimos que, se usarmos apenas um conjunto de codificação com os quinze caracteres mais freqüentes, teremos que colocar os restantes no texto sem codificar, e cada um deles ocupará três nibbles. Entretanto, podemos estender a idéia de um conjunto de quatro bits de código para dois ou mais conjuntos adicionais. Assim, a codificação de cada caractere constante do segundo conjunto ocupará dois nibbles (um com o 0 e outro com o código propriamente dito); a codificação dos caracteres do terceiro conjunto ocupará três nibbles, e assim por diante. Vamos fazer as contas de novo, para o mesmo conjunto de cem caracteres:

90 têm códigos de 4 bits = 45 bytes
9 têm códigos de 8 bits = 9 bytes
1 tem código de 12 bits = 2 bytes

TOTAL = 56 bytes

Como você pode notar, conseguimos aumentar nossa eficiência teórica de compressão para 44% (o que representa uma melhoria de 10% em relação ao esquema discutido anteriormente).

Podemos agora desenvolver um programa de codificação (acrescente as linhas **DATA** de teste ao final):



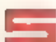
```
10 DIM F$(50),K$(3)
20 K$(1)=" AEOSRDIUNCMTPL"
25 K$(2)=",H.VGBF'Q:!-KZJ"
27 K$(3)="XWY;=/*?()$%+"
180 PRINT "CODIFICANDO..."
190 RESTORE:I=0:NC=0:K=1
200 READ L$:IF L$="*" THEN 230
210 GOSUB 710:I=I+1:F$(I)=X$
212 GOSUB 820
215 NC=NC+LEN(X$)
220 GOTO 200
230 NL=I:T=0
235 PRINT "TOTAL: " ; NC ;
"CARACTERES"
237 STOP
700 REM --- CODIFICACAO
710 N=0:X$="":Y$="":L$=L$+" "
720 FOR J=1 TO LEN(L$)
730 C$=MID$(L$,J,1)
740 P=INSTR(K$(K),C$)
760 N=N+1:IF N=1 THEN B=P
770 IF N=2 THEN N=0:B=B OR
(16*P) : X$=X$+CHR$(B)
775 IF P=0 THEN K=K+1:GOTO 740
776 K=1
780 NEXT J
790 RETURN
```

Coloque um comando **CLEAR 3000** na linha 10, para os micros pertencentes às linhas TRS-80 e TRS-Color.



Para que o programa funcione nos microcomputadores Apple e TK-2000, substitua a rotina de codificação das linhas 710 a 790 por:

```
700 REM --- CODIFICACAO
710 N=0:X$="":L$=L$+" "
720 FOR J=1 TO LEN(L$)
730 C$=MID$(L$,J,1)
735 FOR P=1 TO LEN(K$(K))
740 IF MID$(K$(K),P,1) = C$
    THEN 760
750 NEXT P
760 N=N+1:IF N=1 THEN B=P
770 IF N=2 THEN N=0:B=B OR
    (16*P) : X$=X$+CHR$(B)
775 IF P=0 THEN K=K+1:GOTO 740
776 K=1
780 NEXT J
790 RETURN
```



```
10 DIM f$(50,64),k$(3)
20 k$(1)=" AEOSRDIUNCMTPL"
25 k$(2)=",H.VGBF'Q:!-KZJ"
27 k$(3)="XWY;=/*?()$%+"
180 PRINT "CODIFICANDO..."
190 RESTORE:i=0:nc=0:k=1
200 READ L$:IF L$="*" THEN GOTO
230
210 GOSUB 710:LET i=i+1:LET
f$(i)=x$
212 GOSUB 820
215 LET nc=nc+LEN x$
220 GOTO 200
230 LET nl=I:LET t=0
235 PRINT "TOTAL: ";nc ;
"CARACTERES"
237 STOP
700 REM --- CODIFICACAO
710 LET n=0:LET x$="":LET
L$=L$+" "
720 FOR j=1 TO LEN L$
730 LET c$=L$(j TO j)
735 FOR p=1 TO LEN k$(k)
740 IF k$(K,p TO p) = c$ THEN
GOTO 760
750 NEXT p
760 LET n=n+1:IF n=1 THEN LET
b=p
770 IF n=2 THEN LET n=0:LET b=b
OR (16*p) : LET x$=x$+CHR$ b
775 IF p=0 THEN LET k=k+1:GOTO
740
776 LET k=1
780 NEXT j
790 RETURN
```

As linhas 20 a 27 definem os três conjuntos de caracteres a serem usados. O primeiro conjunto reúne os quinze caracteres mais frequentes em um texto. Para simplificar, utilizaremos a mesma ordem encontrada no texto de teste. Mas, se quiséssemos obter um conjunto padrão, que servisse igualmente bem para qualquer texto na língua portuguesa, deveríamos analisar uma amostra muitas vezes mais extensa antes de definir a ordem dos caracteres. O segundo conjunto contém os próximos quinze caracteres de maior frequência no texto e assim por diante.

A seção do programa que abrange as linhas 200 a 220 lê o texto contido em **DATA** e chama a rotina de codificação, que tem início na linha 710. O texto codificado, devolvido a **X$**, é armazenado no conjunto **F$**, para posterior uso na decodificação.

ROTINA DE CODIFICAÇÃO

Por ser um tanto complexa, a rotina de codificação exige uma explicação mais detalhada. Ela procura, no primeiro conjunto de códigos (que está em **K$(1)**), cada um dos caracteres individuais da linha de texto, extraídos e guardados em **C$**. Não encontrando ali determinado caractere, o apontador **K** é incrementado e a rotina passa a procurá-lo no segundo conjunto, e assim por diante. Todas as vezes que passa para outro conjunto de códigos, a rotina coloca um código 0 na seqüência de saída. Assim, um caractere encontrado no segundo conjunto vai ter um 0 seguido de seu código, e um caractere encontrado no terceiro conjunto terá dois zeros antes de seu código.

A cada dois códigos numéricos gerados (contados pela variável **N**), as linhas 760 e 770 da rotina comprimem os dois nibbles em um byte, **B**, através da expressão matemática da linha 770. Vejamos um exemplo:

	decimal	binário
1º nibble	14	00001110
2º nibble	7	00000111

Expressão: 14 **OR** (16*7)
14 **OR** 112

00001110 **OR** 01110000

que é igual a 01111110 ou 126.

Note que a multiplicação de um nibble por 16 tem o efeito de deslocá-lo da parte baixa para a parte alta do byte. Uma operação **OR** soma o primeiro nibble (não deslocado) com o segundo (deslocado); o resultado é um byte com ambos os nibbles em cada metade.

Se você quiser acompanhar a geração de códigos comprimidos, acrescente as linhas que se seguem e rode o programa novamente



```
212 GOSUB 820
800 REM --- LISTAGEM
820 FOR N=1 TO LEN(X$)
830 PRINT ASC(MID$(X$,N,1));
840 NEXT N:PRINT
850 RETURN
```



```
212 GOSUB 820
800 REM --- LISTAGEM
820 FOR n=1 TO LEN x$
830 PRINT ASC (X$(n TO n));
840 NEXT n:PRINT
850 RETURN
```

Para obter o texto original de volta, basta que você acrescente a rotina de decodificação.



```
237 PRINT "DECODIFICANDO..."
240 FOR I=1 TO NL
250 L$=F$(I):GOSUB 870
260 T=T+1:IF T<15 THEN GOTO 290
270 PRINT:INPUT "PRESSIONE <EN
TER>";X$
280 T=0
290 NEXT I
300 STOP
860 REM --- DECODIFICACAO
870 N=0:K=1:FOR J=1 TO LEN(L$)
880 C=ASC(MID$(L$,J,1))
890 C(1)=C AND 15:C(2)=(C AND
240)/16
```

```
900 FOR L=1 TO 2
910 IF C(L)=0 THEN K=K+1:GOTO
930
920 PRINT MID$(K$(K),C(L),1);
925 K=1
930 NEXT L
940 NEXT J
950 PRINT:RETURN
```

S

```
237 PRINT "DECODIFICANDO..."
240 FOR i=1 TO n1
250 LET L$=f$(i):GOSUB 870
260 LET t=t+1:IF t<15 THEN GOTO
290
270 PRINT:INPUT "PRESSIONE <ENT
ER> ";X$
280 LET t=0
290 NEXT i
300 STOP
860 REM --- DECODIFICACAO
870 LET n=0:LET k=1:FOR j=1 TO
LEN L$
880 LET c=ASC(L$(j TO j))
890 LET c(1)=c AND 15:c(2)=(c
AND 240)/16
900 FOR L=1 TO 2
910 IF c(L)=0 THEN LET
k=k+1:GOTO 930
920 PRINT K$(K,c(L) to c(L));
925 LET k=1
930 NEXT L
940 NEXT J
950 PRINT:RETURN
```

A sub-rotina responsável pela decodificação funciona de maneira exatamente inversa à rotina de codificação, que acabamos de examinar.

A variável **C** contém o código comprimido, extraído de cada um dos caracteres da linha codificada. A partir desta obtêm-se os dois nibbles, que são armazenados respectivamente em **C(1)** e **C(2)**, na expressão da linha 890, e examinados pelo laço das linhas 900 a 930. Se um dos nibbles é igual a 0, o apontador **K**, que inicialmente se encontrava em 1, é incrementado, passando a indicar o próximo conjunto de códigos. Finalmente, a linha 920 imprime na tela o caractere decodificado.

Tomemos o exemplo anterior para ver como funciona a decodificação:

byte = 01111110 = 126

1º nibble 126 **AND** 15

ou: 01111110 **AND** 00001111
ou: 00001110
ou: 14 em decimal

2º nibble (126 **AND** 240)/16

ou: 01111110 **AND** 11110000
ou 01110000
ou: 112/16 = 7 (00000111)

Observe que a divisão por 16 tem o efeito contrário da multiplicação por 16 — ou seja, desloca o nibble alto para o nibble baixo.

GRAU DE EFICIÊNCIA

Obteremos os seguintes resultados com nosso texto de teste: depois de codificado, ele foi reduzido de 1228 para 674 caracteres, ou seja, houve uma redução de 45%. Pouquíssimos métodos de compressão conseguem tão alto grau de eficiência. Na realidade, se contarmos o número total de bytes gastos pelo programa e pela tabela de códigos **K$**, a redução global de espaço vai ser bem menor. Devemos ter em mente, porém, que a utilização de um algoritmo de compressão como este só faz sentido quando o texto a ser comprimido é extenso. Assim, o acréscimo de memória necessário para o programa e as tabelas é insignificante se comparado aos ganhos obtidos pela compressão.

LIÇÕES DA CRIPTOGRAFIA

O compressor de textos desenvolvido aqui toma emprestado alguns conceitos da criptografia, que é a ciência das cifras e dos códigos (veja artigos das páginas 888 e 1091). Ao tentar decifrar um código secreto, o criptógrafo analisa, antes de mais nada, as freqüências dos caracteres presentes no texto cifrado. Estas podem dizer muito sobre a freqüência das letras usadas no idioma original.

O uso das letras mais freqüentes do alfabeto no primeiro conjunto de codificação assegura a obtenção do máximo de compressão possível. Se esse conjunto incluir caracteres que aparecem com menor freqüência, omitindo outros mais utilizados, a compressão perderá muito em eficiência, pois muitos códigos 0 serão gerados.

COMBINANDO ETAPAS

Normalmente, um programa compressor baseado nessa técnica deveria ser independente do texto que se deseja comprimir — ou seja, teria nos conjuntos **K$** uma seqüência padronizada de caracteres mais frequentes, identificados pela análise de uma grande amostra da língua portuguesa.

Entretanto, um texto de aventura apresenta algumas peculiaridades, incluindo termos estranhos, como Aardvark, Niptu, Nordham etc., que não pertencem, evidentemente, à nossa língua. Por isso, verificou-se uma frequência atípica da letra K.

Como precisamos codificar o texto de uma aventura apenas uma vez, nada impede que combinemos os programas de análise de frequências e de codificação em um só, para otimizar o processo. Os programas anteriores foram escritos com números diferentes de linhas, de modo a facilitar essa integração. Para proceder à fusão dos dois programas, faça as modificações:



```
565 K=1
615 K$(K)=K$(K)+
CHR$(C(I)+31)
616 IF LEN(K$(K))=15
THEN K=K+1
```

e suprima as linhas 20 a 27.

S

```
565 LET k=1
615 LET k$(k)=k$(k)+CHR$(c
(i)+31)
616 IF LEN(k$(k))=15 THEN LET
k=k+1
```

e suprima as linhas 20 a 27.

ELEMENTOS DE K$

Observe que algumas linhas foram acrescentadas à rotina de impressão dos resultados da análise de frequência. Elas se encarregam de construir automaticamente os elementos do conjunto **K$**, com a vantagem de criar tantos elementos quantos se façam necessários (se forem mais de três, é conveniente alterar a dimensão deste conjunto na linha 10 do programa principal). Por isso, as linhas 20 a 27, que contêm a definição fixa do conteúdo do conjunto **K$**, devem ser retiradas.

Tente codificar outros textos com esse programa, e veja como a taxa de compressão se mantém quase a mesma. Esta é uma característica desejável de qualquer sistema de compressão.

No próximo artigo sobre este tema, investigaremos outras alternativas utilizadas na compressão de textos, particularmente o *Método Chinês*, por meio do qual é possível obter 60% de redução em uma aventura.



OS SEGREDOS DO SPECTRUM (2)

- VARIÁVEIS DO SISTEMA
- TIPOS DE VARIAVEL
- COMO MODIFICAR
- APONTADORES E CONTADORES
- ENDERECOS UTEIS

Para o programador ambicioso, existe um mapa de valor inestimável: os endereços das variáveis do sistema. Com PEEK e POKE, você terá acesso a ele e obterá maravilhas com o micro.

O BASIC do Spectrum possui um grande numero de comandos relativamente poderosos para trabalho com a tela, teclado, memória etc. Apesar disso, muitas propriedades da máquina e de seu sistema operacional ficam fora do alcance do programador que se restringe apenas a esses comandos.

Usando a função **PEEK** para leitura direta de uma locação absoluta de memória, e o comando **POKE** para modificar seu conteúdo, você poderá redefinir muitos parâmetros de funcionamento do sistema operacional e do interpretador BASIC no Spectrum.

Mas, se o sistema operacional e o interpretador estão gravados permanentemente na memoria ROM da máquina, como é possível alterar o seu funcionamento? Quando o computador e ligado, o interpretador BASIC especifica uma área de trabalho na RAM, que vai da locação 23522 a 23~32. Uma area adicional de 23734 a 23791 e criada, se o computador for conectado a uma interface universal de entrada/saida.

Cada locação nessas áreas armazena um valor numerico, usado pelo sistema operacional e pelo interpretador durante a execução de programas. Essas locações são chamadas *variáveis do sistema* e recebem um nome simbólico, para fins de referência. Podem ser:

- registros intermediários de operações, tais como: código da última tecla pressionada (**LASTK**), último código de erro encontrado (**ERRNR**) etc,
- parâmetros de operação de comandos: duração do bipe (**RASP**), atraso entre repetições de uma tecla (**REPPER**), cor da moldura (**BORDCR**) etc;
- indicadores: próxima linha a ser executada (**NEWPPC**), coordenada X do cursor de alta resolução (**COORDX**), número de linha da posição corrente de **PRINT** (**SPOSLN**) etc;
- apontadores, que contêm outros endereços de memória: início de um programa BASIC (**PROG**), endereço do cursor na RAM da tela (**KCUR**) etc;
- parâmetros de definição do hardware: topo da memória RAM (**PRAMT**) etc;
- contadores e acumuladores: contador de quadros de TV (**FRAMES**), espaço ocupado por cadeias (**STRLEN**) etc.

Algumas variáveis do sistema ocupam um byte, outras, dois. Quando ocupam dois bytes, o número de dezesseis bits nelas armazenado segue o esquema de dupla byte mais significativo/byte menos significativo.

VARIÁVEIS DO SISTEMA

Nome	Endereço	N° de bytes	Utilização
LASTK	23560	1	Última tecla pressionada
REPDEL	23561	1	Tempo de pressão a uma tecla para repetir
REPPER	23562	1	Atraso entre repetições de uma tecla
RASP	23608	1	Duração do som de alarma
PIP	23609	1	Duração do "clique" de teclado
ERRNR	23610	1	Último código de erro, menos 1
MODE	23617	1	Modo do cursor (K, L, C, E ou G)
NEWPPC	23618	2	Próxima linha a ser executada
NSPPC	23620	2	Próximo comando da linha a ser executada
EPPC	23625	2	Número da linha com o cursor do programa
VARS	23627	2	Endereço do início da área de variáveis
PROG	23635	2	Endereço do início do programa BASIC
NXTLIN	23637	2	Endereço da próxima linha a executar
ELINE	23641	2	Endereço do *buffer* de entrada em BASIC
KCUR	23643	2	Endereço do cursor
FLAGS2	23658	1	Indicador de minúsculas/maiúsculas
DFSZ	23659	1	Número de linhas na zona inferior da tela
SEED	23670	2	Semente do gerador **RAND**
FRAMES	23672	3	Contador de quadros de TV
COORDX	23677	1	Coordenada X do último ponto gráfico
COORDY	23678	1	Coordenada Y do último ponto gráfico
PPOSN	23679	1	Coluna de impressão na impressora
FREE	23681	1	Byte livre para o usuário, não afetado por **NEW**
SPOSN	23688	1	Número de coluna da última posição **PRINT**
SPOSLN	23689	1	Número da linha da última posição **PRINT**
SCRCT	23692	1	Contador de linhas para o *scroll*
RAMTOP	23730	2	Topo da memória livre para BASIC
PRAMT	23732	2	Topo da memória RAM (física)

APLICAÇÕES

O acesso a esses endereços permite-nos, entre outras coisas, copiar determinado valor em uma variável do sistema, de modo a informar o programa sobre algum parâmetro de interesse para o processamento subseqüente. Suponhamos que você queira dimensionar uma tabela em função do máximo de memória disponível no computador. Para isso, use o número de dezesseis bits armazenado no par 23730-23731, que é a variável de sistema **RAMTOP**. Ela indicará o topo máximo de memória RAM disponível para um programa em BASIC:

```
LET MAX = PEEK(23730)+PEEK(
23731)*256
```

Podemos também modificar um valor por meio de um **POKE**, fazendo com que dado comando seja executado de maneira diferente. Por exemplo: a locação 23692 corresponde à variável do sistema **SCRCT**, que é o contador de linhas usado pelo sistema para perguntar *scroll?* quando o cursor de texto chega ao fim da tela. Com um **POKE 23692,255** a cada número de linha, o computador nunca fará essa pergunta.

No quadro acima, listamos algumas das variáveis de sistema mais importantes para o Spectrum e compatíveis. Use a sua imaginação!

LOGO: ALÉM DA TARTARUGA

- EXPRESSÕES MATEMÁTICAS
- VARIÁVEIS E NÚMEROS
- NOMES E SENTENÇAS
- O QUE SÃO LISTAS
- COMANDOS PARA LISTAS

Os recursos do LOGO não se esgotam em seus eficientes comandos gráficos. Manipular palavras, representar dados, efetuar cálculos matemáticos — tudo isso o LOGO pode, e muito mais.

[illegible] programação LOGO [illegible] naremos as características fundamentais dessa linguagem. Na realidade, seguimos um caminho diverso do que é usual no aprendizado de outras linguagens: já que abordamos em primeiro lugar os comandos gráficos. Mas esse é o meio mais fácil de entrar em contato com o [illegible] LOGO é apenas uma "linguagem gráfica", própria para crianças.

[illegible] Como o LISP, seu antecessor, o LOGO foi projetado de modo a simplificar a manipulação de palavras e listas de palavras e frases. Essa característica não impede, contudo, que complexos programas de inteligência artificial sejam desenvolvidos com a ajuda do LOGO. [illegible] BASIC [illegible]







O que é um micromundo LOGO?

Um micromundo é um conjunto conexo de procedimentos, escritos em LOGO, que estruturam uma determinada situação ou ambiente de resolução de problemas cognitivos.

Esses procedimentos podem ser combinados em outros procedimentos, de forma cada vez mais complexa, imitando assim o desenvolvimento natural do *conhecimento* da criança sobre uma parte do mundo.

Exemplificando: podemos construir um micromundo em que a tela é a gaiola de um coelhinho, com lugares para ele brincar, dormir, comer etc. Usando o LOGO, a criança e o professor elaboram vários procedimentos para desenhar o ambiente e seus objetos, fazer o coelhinho se movimentar, levá-lo até a área de dormir etc., de maneira progressiva e hierárquica. Esse micromundo pode ser tão complexo e completo quanto se queira.

(constantes e variáveis), os comandos de impressão etc. É o que faremos agora com a linguagem LOGO.

Como nos artigos anteriores, apresentaremos sempre duas versões dos programas: a primeira, identificada pelo logotipo Apple, corresponde ao padrão MIT LOGO, em inglês, adotado universalmente com poucas variações; a segunda, identificada pelo logotipo do MSX, corresponde à versão nacional, chamada BRASLOGO, desenvolvida pela Unicamp para a Itautec e adotada por outros fabricantes. Ao final do artigo, damos a tradução dos comandos usados para o MLOGO, versão nacional muito difundida, destinada ao Apple.

MATEMÁTICA EM LOGO

Como qualquer linguagem de programação que se preze, o LOGO dispõe de amplos recursos destinados a cálculos matemáticos com diferentes níveis de complexidade; não se restringe às quatro operações aritméticas.

O LOGO não faz distinção entre números inteiros e reais (números que podem ser fracionários, como 1.2, 3.14156 etc.). Em alguns interpretadores, entretanto, a falta do ponto decimal nos argumentos de uma operação força um resultado inteiro.

Os operadores matemáticos são iguais aos da linguagem BASIC: soma (+), subtração (−), divisão (/), multiplicação (*), parênteses etc. A ordem de execução das operações também é a mesma. Usando, por exemplo, o comando **PRINT (ESCREVA**, na versão em português), com função idêntica à que tem no BASIC, podemos fazer:



```
PRINT 2 + 2*3
```



```
ESCREVA 2 + 2*3
```

O resultado da operação será igual a 8, pois, nesta expressão, efetua-se a multiplicação em primeiro lugar.

As expressões matemáticas podem ser colocadas em outros comandos do LOGO, como no desenho de gráficos:



```
FORWARD (10 + 120)/3.14
```



```
PARAFRENTE (10 + 120)/3.14
```

Os procedimentos também comportam expressões matemáticas:



```
TO PI2
PRINT 3.14156*2
END
```



```
APRENDA PI2
ESCREVA 3.14156*2
FIM
```

Feito isso, sempre que digitarmos o comando **PI2** pelo teclado, obteremos a seguinte resposta:

```
2.28312
```

Uma expressão pode utilizar variáveis. Como você deve estar lembrado, as variáveis recebem nomes, que podem ser de qualquer comprimento, e são identificadas por dois pontos. Assim, um procedimento destinado a mostrar o cubo de um número qualquer seria escrito da seguinte maneira:



```
TO CUBO :NUMERO
PRINT :NUMERO*:NUMERO*:NUMERO
END
```



```
APRENDA CUBO :NUMERO
ESCREVA :NUMERO*:NUMERO*:NUMERO
FIM
```

O procedimento criado exige um argumento numérico de entrada, e responde imprimindo na tela o cubo do mesmo:

```
CUBO 3
27
CUBO 10
1000
```

Eis aqui outro procedimento, que calcula a media de dois numeros:



```
TO MEDIA :N1 :N2
PRINT (:N1 + :N2)/2
END
```







```
APRENDA MEDIA :N1 :N2
ESCREVA (:N1 + :N2)/2
FIM
```

Esse procedimento requer a presença de dois argumentos:

```
MEDIA 2 4
3
MEDIA 1 2
1.5
```

Em alguns interpretadores LOGO, a resposta ao segundo comando poderá ser igual a 1, ou seja, o resultado será mostrado em forma inteira, pois os argumentos de entrada estavam expressos desse modo. Se isso ocorrer, podemos forçar uma operação com resultados fracionários, escrevendo:

```
MEDIA 1.0 2.0
```

Não se esqueça de colocar os dois argumentos para o procedimento **MEDIA**. Caso contrário, o interpretador reclamará, exibindo uma mensagem de erro.

Observe que, em todas as expressões aritméticas examinadas, o símbolo da operação matemática a ser realizada aparece entre os argumentos da operação (por exemplo, 2+2). Essa notação, chamada *infixa*, é a mais usada em linguagens imperativas derivadas do FORTRAN. Mas existem dois outros tipos de notação: a *sufixa*, ou notação polaca inversa (RPN), que é usada em linguagens como o FORTH (a mesma operação acima seria escrita 2 2 +), e a *prefixa*, que também é aceita pela linguagem LOGO. Nesse tipo de notação, o comando precede os argumentos que usa.

Para realizar operações prefixas no LOGO, empregam-se comandos já existentes no interpretador. Por exemplo:



```
SUM 6 7
DIVIDE 3 1.2
PRODUCT 3 4
```

Em alguns interpretadores LOGO, esses comandos recebem nomes diferentes, como **QUOTIENT** em vez de **DIVIDE**, ou **MULTIPLY** em vez de **PRODUCT**. Pode acontecer, também, que o interpretador não contenha o procedimento **SUBTRACT**. Nesse caso, basta usar **SUM**, com um número negativo.



```
SOMA 6 7
QUOCIENTE 3 1.2
PRODUTO 3 4
```

Se um desses comandos não existir no interpretador LOGO que está usando, deverá fazê-lo **APRENDER** o novo procedimento.

Note que é possível utilizar funções dentro de funções, como em:



```
DIVIDE SUM 3 2 SUM 4 5
```



```
QUOCIENTE SOMA 3 2 SOMA 4 5
```

que corresponde à notação infixa:

```
PRINT (3+2)/(4+5)
```

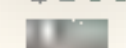
Não tente fazer o mesmo, porém, com os procedimentos **CUBO** e **MEDIA**, desenvolvidos anteriormente, pois eles apenas imprimem os respectivos resultados, não tendo meios de "passá-los" a uma outra função. Experimente, por exemplo, digitar a seguinte expressão para ver o que acontece:

```
CUBO MEDIA 3 4 7 88
```

Para retornar um resultado, o procedimento precisa incluí-lo na lista de variáveis na chamada. Uma alternativa mais simples consiste em usar o comando **OUTPUT** (ou **ENVIE**, na versão para a língua portuguesa):



```
TO MEDIA :N1 :N2
OUTPUT (:N1 + :N2)/2
```



```
APRENDA MEDIA :N1 :N2
ENVIE (:N1 + :N2)/2
FIM
```

Agora, você poderá recorrer ao procedimento **MEDIA** como argumento de outro procedimento.

FUNÇÕES MATEMÁTICAS

O LOGO possui várias funções matemáticas que facilitam o cálculo de expressões mais complexas — algébricas, trigonométricas etc.



REMAINDER	Resto de uma divisão
ROUND	Arredondamento
ABS	Absoluto
INT	Inteiro
SQRT	Raiz quadrada
SIN	Seno
COS	Co-seno



RESTO	Resto de uma divisão
INTEIRO	Inteiro
RAIZQ	Raiz quadrada
SENO	Seno
COS	Co-seno

Só alguns interpretadores têm o conjunto completo dessas funções.

O LOGO conta ainda com um gerador de números aleatórios, útil em uma série de aplicações, inclusive jogos.

Eis um programa que simula dez lançamentos de um dado:



```
TO DADOS
REPEAT 6 [PRINT RANDOM 6]
```





```
TO DADOS
REPITA 6 [ESCREVA SORTEIEATE 6]
FIM
```

Para inicializar uma sequência aleatória, utiliza-se o comando **RANDOMIZE** (**REPRODUZA**, na versão BRASLOGO).

PALAVRAS

É possível especificar palavras (constantes alfanuméricas) em LOGO, identificando-as com o serial de aspas. Se quiser, por exemplo, imprimir no vídeo a palavra COMPUTADOR, digite:



```
PRINT "COMPUTADOR
```



```
ESCREVA "COMPUTADOR
```

Note que não se fecham as aspas, como no BASIC. Se você se esquecer de colocar as aspas antes da palavra, a máquina entenderá que COMPUTADOR é o nome de um número ou de procedimento criado anteriormente; não o encontrando, exibirá uma mensagem de erro.

Como o comando de impressão aceita apenas um argumento, as formas que se seguem também são incorretas:



```
PRINT "BOM DIA
PRINT "BOM "DIA
```



```
ESCREVA "BOM DIA
ESCREVA "BOM "DIA
```

No primeiro caso, o interpretador LOGO informará que ainda não conhece DIA. No segundo, dirá que não sabe o que fazer com "DIA (ou, dependendo do computador, que há um argumento a mais). É importante lembrar que um espaço em branco sempre indica, em linguagem LOGO, o fim de uma palavra, e que não podemos colocar mais de uma palavra após um sinal de aspas.

Para imprimir duas palavras usando um mesmo comando, precisamos antes juntá-las em um único argumento. Existe um comando primitivo que faz isso:



```
PRINT WORD "BOM "DIA
```



```
ESCREVA PALAVRA "BOM "DIA
```

O comando **WORD** (ou **PALAVRA**) toma dois argumentos — que são letras ou palavras precedidas de aspas — e os reúne em uma só palavra. Esta é passada para o comando **PRINT** (ou **ESCREVA**), que então a coloca na tela.

O LOGO possui diversos comandos extremamente poderosos para lidar com palavras. Para extrair letras de uma palavra, utilizam-se estes comandos:

MICRO DICAS

TRADUÇÃO PARA O MLOGO

Os comandos do MLOGO, para o Apple, correspondentes aos comandos do BRASLOGO deste artigo, são:

BRASLOGO	MLOGO
ESCREVA	MOSTRAR
ATRIBUA	FACA
COLOQUE	—
SOMA	SOME
PRODUTO	PRODUTO
QUOCIENTE	QUOC
RESTO	RESTO
RAIZQ	RQD
INTEIRO	INT
—	APROX
REPRODUZA	RESORTEIE
SORTEIEATÉ	SORTEIE
SEN	SEN
COS	COS
ARCTAN	ATAN
PALAVRA	PALAVRA
SENTENÇA	SENTENÇA
PRIMEIRO	PRIMEIRO
SEMPRIMEIRO	SEMPRIMEIRO
ULTIMO	ULTIMO
SEMÚLTIMO	SEMULTIMO
ENVIE	SAIDA





FIRST - Extrai a primeira letra.
LAST - Extrai a última letra.
BUTFIRST - Extrai todas as letras, menos a primeira.
BUTLAST - Extrai todas as letras, menos a última.

Como exemplo, tente os comandos:

```
PRINT FIRST "ABCD
PRINT LAST "ABCD
PRINT BUTFIRST "ABCD
PRINT BUTLAST "ABCD
PRINT FIRST BUTFIRST "ABCD
PRINT LAST BUTLAST "ABCD
PRINT WORD FIRST "ABCD LAST
   "ABCD
```

Note como os comandos são encadeados: **FIRST BUTFIRST** extrai a segunda letra da palavra ABCD; **LAST BUTLAST** extrai a penúltima palavra etc.

O pequeno procedimento recursivo que mostramos a seguir é capaz de extrair todas as letras de uma palavra:

```
TO LETRAS :PALAVRA
IF :PALAVRA = " THEN STOP
PRINT FIRST :PALAVRA
LETRAS BUTFIRST :PALAVRA
END
```

O procedimento ilustra o conceito de *palavra vazia* — ou seja, palavra sem nenhum caractere. A chamada recursiva de **LETRAS**, na quarta linha, usa como argumento o **BUTFIRST** da palavra que foi entrada. Esta vai sendo reduzida até não ter mais nenhuma letra. A recursão só se encerra quando o teste executado pelo **IF** da segunda linha é verdadeiro. A palavra vazia é assinalada pelas aspas e um espaço em branco.



PRIMEIRO - Extrai a primeira letra.
ÚLTIMO - Extrai a última letra.
SEMPRIMEIRO - Extrai todas as letras, menos a primeira.
SEMÚLTIMO - Extrai todas as letras, menos a última.

Como exemplo, tente os comandos:

```
PRINT PRIMEIRO "ABCD
PRINT ULTIMO "ABCD
PRINT SEMPRIMEIRO "ABCD
PRINT SEMULTIMO "ABCD
PRINT PRIMEIRO SEMPRIMEIRO
   "ABCD
PRINT ULTIMO SEMULTIMO "ABCD
PRINT PALAVRA PRIMEIRO "ABCD
ULTIMO "ABCD
```

Note como os comandos são encadeados: **PRIMEIRO SEMPRIMEIRO** extrai a segunda letra da palavra ABCD; **ÚLTIMO SEMÚLTIMO** extrai a penúltima palavra e assim por diante.

O pequeno procedimento recursivo mostrado a seguir é capaz de extrair todas as letras de uma palavra:

```
APRENDA LETRAS :PALAVRA
SE :PALAVRA = " ENTAO [PARE]
ESCREVA PRIMEIRO :PALAVRA
LETRAS SEMPRIMEIRO :PALAVRA
FIM
```

O procedimento ilustra o conceito de *palavra vazia* — ou seja, palavra sem nenhum caractere. A chamada recursiva de **LETRAS**, na quarta linha, usa como argumento o **SEMPRIMEIRO** da palavra que foi entrada. Esta vai sendo reduzida até não ter mais nenhuma letra. A recursão só se encerra quando o teste executado pelo **SE** da segunda linha é verdadeiro. A palavra vazia é assinalada pelo sinal de aspas seguido de um espaço em branco.

ATRIBUIÇÃO

O comando de atribuição — **LET**, em linguagem BASIC — também existe em LOGO. Com ele, podemos armazenar números ou resultados de expressões em variáveis, e dar-lhes nome.



```
MAKE "IDADE 15
```



```
ATRIBUA "IDADE 15
```

Podemos ainda colocar uma palavra em variável:



```
MAKE "NOME "JOAQUIM
```



```
ATRIBUA "NOME "JOAQUIM
```

Experimente verificar o conteúdo das variáveis **NOME** e **IDADE**, com:



```
PRINT :NOME
PRINT :IDADE
```



```
ESCREVA :NOME
ESCREVA :IDADE
```

Como você provavelmente deve ter observado, usamos dois pontos, e não aspas, quando estamos nos referindo a uma variável já criada. Esta é uma característica fundamental do LOGO. Nessa linguagem, ao contrário do BASIC, é necessário diferenciar *explicitamente* as referências à variável (**"NOME**) e ao seu conteúdo (**:NOME**).

Na realidade, todos os elementos do LOGO são definidos do mesmo modo, e armazenados na mesma estrutura. Tanto os comandos (programas) quanto os dados são *palavras*. Números também são tratados como palavras. Assim, é perfeitamente possível fazer-se algo como:



```
PRINT "2.34 + "19.87
PRINT BUTFIRST 12.345
```



```
ESCREVA "2.34 + "19.87
ESCREVA SEMPRIMEIRO 12.345
```

Eis aqui um pequeno programa que ilustra essa característica do LOGO. Ele imprime uma tabela contendo o quadrado e a raiz quadrada de todos os números inteiros, de **N1** a **N2**:



```
TO TABELA :N1 :N2
IF :N1=:N2 THEN STOP
PRINT WORD :N1 :N1*N1 SQRT(:N1)
TABELA :N1+1 :N2
END
```



```
TO TABELA :N1 :N2
SE :N1=:N2 ENTAO [PARE]
ESCREVA PALAVRA :N1 :N1*N1
RAIZQ(:N1)
TABELA :N1+1 :N2
FIM
```

Para fazer o procedimento **TABELA** chamar a si mesmo, usamos a recursão incrementando a variável de início **:N1**. A segunda linha do procedimento verifica o valor dessa variável, parando se ela já tiver atingido o máximo desejado (**:N2**). Caso contrário, imprime na tela uma linha contendo o número, seu quadrado e sua raiz quadrada. Como o comando de escrita aceita um só argumento, precisamos usar um "aglutinador" dos diferentes números, colocando-os em uma única lista, por meio do comando **WORD** (ou **PALAVRA**).

LISTAS

Todo o potencial do LOGO se revela quando examinamos a última de suas estruturas básicas de dados: a *lista*, um conjunto de palavras indicado por meio de colchetes. Por exemplo:

```
[TIGRE LEAO GATO]
```

é uma lista com três elementos, separados por brancos.

Podemos dar nome às listas:

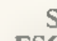


```
MAKE "FELINOS [TIGRE LEAO GATO]
MAKE "IMPARES [1 3 5 7 9]
MAKE "COLEGAS [JOAQUIM 15 MARIO
14]
```



```
ATRIBUA "FELINOS [TIGRE LEAO
GATO]
ATRIBUA "IMPARES [1 3 5 7 9]
ATRIBUA "COLEGAS [JOAQUIM 15
MARIO 14]
```

Se usarmos o comando **PRINT** (ou **ESCREVA**) seguido do nome de uma lista, obteremos seu conteúdo completo. Uma lista pode conter outras listas:



```
MAKE "CARNIVOROS [FELINOS LOBO
RAPOSA]
```



```
ATRIBUA "CARNIVOROS [FELINOS
LOBO RAPOSA]
```

Todas as funções utilizadas para manipulação de palavras aplicam-se também a listas. Por exemplo:



```
PRINT FIRST FELINOS
```



```
ESCREVA PRIMEIRO FELINOS
```

Isso fará com que o primeiro elemento da lista, que é TIGRE, seja impresso na tela.

Uma lista pode ter apenas um elemento, ou nenhum. Nesse caso, denomina-se *lista vazia*, e é representada pelo símbolo [ ]. Para unir duas listas, usamos o comando primitivo **SENTENCE** (ou **SENTENÇA**):



```
PRINT SENTENCE COLEGAS IMPARES
```



```
ESCREVA SENTENÇA COLEGAS
IMPARES
```

Procuramos aqui demonstrar a riqueza e a versatilidade do LOGO. Esperamos ter motivado o leitor a continuar explorando essa poderosa linguagem.



Quais as etapas para o desenvolvimento de uma nova linguagem?

Desenvolver uma nova linguagem não é difícil — tanto que existem centenas delas que nunca passaram do estágio de mera "brincadeira" científica para uma efetiva implementação e distribuição no mercado.

Em primeiro lugar, deve-se definir se a linguagem será imperativa, funcional, procedimental etc. e que tipo de comandos, funções e instruções terá. Depois, é preciso decidir se será uma linguagem *interpretada*, *compilada*, ou ambas.

Finalmente, escreve-se o programa — geralmente em Assembler, C, ou qualquer outra linguagem otimizada para o desenvolvimento de ferramentas de software — que irá realizar o processo de tradução da nova linguagem em códigos de máquina destinados ao computador escolhido.

As técnicas para a elaboração do compilador ou interpretador, complexas e altamente especializadas, são ensinadas em cursos superiores de análise de sistemas.



FUNÇÕES PODEROSAS

- MAIOR MÚLTIPLO
- MENOR MÚLTIPLO
- RESTO DE UMA DIVISÃO
- ARREDONDAMENTO
- PAR OU ÍMPAR?

O comando DEF FN permite incorporar ao elenco de funções matemáticas do BASIC um conjunto adicional de cálculos de grande interesse para o programador. Utilize-o.

No artigo da página 608, vimos como definir novas funções e usá-las em diversos tipos de manipulação de dados. Trataremos aqui de certas funções matemáticas úteis ao programador.

Um extenso grupo de funções matemáticas refere-se aos *módulos*, números que "cabem" dentro de outros números, ou que restam de uma divisão.

Há várias aplicações para esses cálculos. Tomemos como exemplo uma questão corriqueira, que envolve a categoria mais geral dos múltiplos: qual é o primeiro múltiplo de 100, menor ou igual a 345? Não é preciso ser nenhum gênio para chegar à resposta — 300, que é o *arredondamento* do número 345 para baixo, até a centena mais próxima. Mas qual é o primeiro múltiplo de 64, menor ou igual a 511? Eis uma função que resolve esse problema:



```
DEF FNMN(N1,N2)=INT(N1/N2)*N2
```

O segundo argumento corresponde ao número cujo múltiplo queremos achar; o primeiro, ao valor máximo desse múltiplo. Nos exemplos dados, poderíamos usar: **PRINT FNMN(345,100)** e **FNMN(511,64).** Como o Apple, o TK-2000 e o TRS-Color não admitem mais que um argumento por função, a linha anterior não pode ser executada nessas máquinas. A solução é utilizar só um argumento (veja o artigo da página 608).

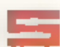
Também interessante é a função que determina o primeiro múltiplo de um número, superior a um certo valor. Exemplo: o primeiro múltiplo de 100 maior que 3 022 é 3 100.



```
DEF FNMM(N1,N2)=INT(N1/N2)*N2+
    N2
```

Os argumentos têm o mesmo significado que os da função anterior.

Existem ainda outros tipos de arredondamento. O mais comum consiste em arredondar os dígitos da parte fracionária de um número usando a "regra do 5": se o dígito a desprezar é maior ou igual a 5, o que está à sua esquerda é arredondado para cima; caso contrário, este não muda. A função que executa esse tipo de arredondamento é:



```
DEF FNAR(N,D)=INT(N*10**D)/10**
D
```



```
DEF FNAR(N,D)=INT(N*10↑D)/10↑D
```



```
DEF FNAR(N,D)=INT(N*10^D)/10^D
```

A função tem dois argumentos: **N**, o número a arredondar, e **D**, o número de decimais desejado. Se quisermos, por exemplo, arredondar para dois decimais o número 23.4567, especificamos **PRINT FNAR(23.4567,2).**

Também é útil a função de arredondamento que determina o próximo número inteiro a partir de um número fracionário. É uma função diferente de **INT** (que obtém a parte inteira de um número, arredondando em alguns casos), ou de **FIX** (que determina apenas a parte inteira, sem arredondar):



```
DEF FNRD(N)=FIX((FIX(N*10)+SGN
    (N)*5)/10)
```



```
DEF FNRD(N)=INT((INT(N*10)+SGN
    (N)*5)/10)
```

Outra função importante é a que calcula o resto inteiro de uma divisão entre dois números **N1** e **N2**.



```
DEF FNRE(N1,N2)=N1-FNMN(N1,N2)
```

Utilizamos aqui a função **FNMN**, já definida. A função **FNRE** pode ser empregada quando se quer determinar se um número é par ou ímpar.



```
DEF FNPI(N)=2-INT(N/2)*2
```



```
DEF FNPI(N)=FNRE(N,2)
```

Se o resultado da função for 1, o número é ímpar; se for 0, é par.

O menu deste programa permite testar as funções apresentadas:



```
25 DEF FNMM(N1,N2)=INT(N1/N2)*N
2+N2
30 DEF FNMN(N1,N2)=INT(N1/N2)*N
2
35 DEF FNRE(N1,N2)=N1-FNMN(N1,N
2)
40 DEF FNRD(N)=FIX((FIX(N*10)+S
GN(N)*5)/10)
50 DEF FNPI(N)=FNRE(N,2)
60 DEF FNAR(N,D)=INT(N*10^D)/10
^D
100 CLS
110 PRINT "DEMONSTRACAO DE FUNC
OES MATEMATICAS"
120 PRINT
130 PRINT "(1)   PRIMEIRO MULTI
PLO MAIOR"
140 PRINT "(2)   PRIMEIRO MULTI
PLO MENOR OU IGUAL"
150 PRINT "(3)   RESTO DE UMA D
IVISAO"
160 PRINT "(4)   ARREDONDAMENTO
 DE UM NUMERO"
165 PRINT "(5)   ARREDONDAMENTO
PARA O MAIOR INTEIRO"
170 PRINT "(6)   PAR OU IMPAR"
180 PRINT "(7)   FIM"
190 PRINT
200 INPUT "OPCAO ";OP
210 IF OP=7 THEN STOP
220 IF OP>3 THEN INPUT "ARGUMEN
TO ";N:GOTO 250
230 INPUT "PRIMEIRO ARGUMENTO "
;N1
235 INPUT "SEGUNDO ARGUMENTO ";
N2
240 IF OP=1 THEN X=FNMM(N1,N2)
250 IF OP=2 THEN X=FNMN(N1,N2)
260 IF OP=3 THEN X=FNRE(N1,N2)
270 IF OP=4 THEN X=FNAR(N)
275 IF OP=5 THEN X=FNRD(N)
280 IF OP=6 THEN X=FNPI(N)
290 PRINT "RESULTADO = ";X
300 GOTO 120
```

PAPEL, PEDRA, TESOURA

Quer dizer que você acredita ser fácil blefar um computador? Experimente estes programas: eles mudarão sua "arrogante" opinião, colocando-o diante de um imbatível adversário.



- JOGOS DE BLEFE
- ALEATORIEDADE
- COMO CLASSIFICAR AS POSSIBILIDADES
- AS REGRAS DO JOGO
- O USO DA ESTATÍSTICA
- O QUE É ALISAMENTO EXPONENCIAL
- TÉCNICA DE SOMA CUMULATIVA



Um computador que possa enfrentar um blefe não é coisa do futuro: já existe aqui e agora. Com os programas em BASIC deste artigo, não há escapatória: a máquina será implacável com quem ousar desafiá-la em um clássico jogo de blefe: *Papel, Pedra, Tesoura*.

É UMA PEDRADA

Jogos de blefe são todos aqueles em que ambos os participantes mostram suas jogadas ao mesmo tempo, não havendo nenhuma informação prévia sobre a probabilidade da jogada de cada um. Um exemplo clássico é *Papel, Pedra, Tesoura*, no qual dois jogadores representam com gestos a forma de uma pedra, de uma folha de papel ou de uma tesoura. A escolha de cada um é revelada simultaneamente, como em um jogo de par ou ímpar. Ganha a rodada aquele que mostrar o objeto mais poderoso, de acordo com as seguintes regras:

- tesoura corta papel, portanto, tesoura ganha;
- papel embrulha pedra, portanto, papel ganha;
- pedra quebra tesoura, portanto, pedra ganha.

Se ambos os jogadores escolherem o mesmo objeto, haverá empate. Joga-se um número predeterminado de rodadas — vinte, por exemplo — e conta-se um ponto para cada rodada ganha.

Neste artigo, desenvolveremos duas versões de um programa que joga *Papel, Pedra, Tesoura* muito bem. Tão bem que você até se sentirá um pouco abalado quando perceber que o computador parece adivinhar suas intenções...

MAIS SÔBRE BLEFE

Como um programa é capaz de predizer eventos aparentemente aleatórios? Uma das "saídas" seria a simples adivinhação — ou seja, o programa sortearia números ao acaso (digamos, o número 1 representaria o papel, o 2, a pedra, e o 3, a tesoura), tanto para determinar a jogada do computador, quanto para tentar adivinhar a do oponente. Um jogo como este não é satisfatório do ponto de vista estratégico: com um número de rodadas grande, certamente se chega a um empate.

Para sorte do computador, entretanto, o homem não age de maneira puramente aleatória, tendendo a exibir com maior freqüência um ou dois tipos de objetos. Quase sempre, porém, os padrões de apresentação são bem mais complexos, seguindo ciclos ou mudando gradativamente as probabilidades.

Além de sua incapacidade natural de atuar como um gerador perfeito de números aleatórios, o jogador humano tem mais um ponto vulnerável: a compulsão a responder ao que acontece no jogo. Quer esteja ganhando, quer esteja perdendo, é quase certo que formulará "teorias" — apelando, via de regra, para a superstição — sobre quais são as respostas com maior chance de garantir-lhe a vitória. A distração proporcionada pelo jogo também tende a reduzir a frieza matemática de um jogador, mesmo que ele lute contra isso.

De qualquer forma, não resta dúvida de que um jogador humano, por ser humano, tentará ampliar seus ganhos ao máximo de alguma estratégia. Isso significa que haverá sempre um viés, ou vício de jogada, que, se cuidadosamente analisado, poderá dar indícios de como será seu próximo lance.

Essa análise poderia ser feita por meio de cálculos estatísticos de diversos tipos. Mas o computador é bem mais eficiente no que se refere a cálculos — e é isto o que lhe permitirá sair vitorioso em um jogo de blefe.

Se um jogador modifica sua estratégia gradualmente, a técnica estatística mais adequada é a do *alistamento exponencial*. Se, ao contrário, o oponente muda de tática rapidamente, a melhor técnica é a da *soma cumulativa*.

COMO JOGAR

Para enfrentar o computador em *Papel, Pedra, Tesoura*, você precisa simplesmente pressionar a tecla 1, 2 ou 3, quando chegar a sua vez.

É importante notar o seguinte: em-

bora a máquina tenha uma grande chance de vencê-lo, ela não frauda o jogo, "olhando" sua jogada antes de decidir a dela. Na verdade, o método utilizado consiste em fazer o computador analisar as escolhas já feitas e decidir *antes* que você entre uma nova jogada.

ALISAMENTO EXPONENCIAL

O alisamento exponencial é a técnica estatística empregada para regularizar o contorno de uma curva cheia de altos e baixos. Se você colocar em um gráfico o número de pedras, tesouras e papéis que o oponente apresentou em cada unidade de tempo (por exemplo, a cada dez jogadas), a curva terá o aspecto de uma serra. Porém, regularizando o contorno da curva — ou seja, tornando as subidas e descidas mais "suaves" —, o computador detectará alguma tendência significativa a longo prazo, como, por exemplo, um aumento expressivo na apresentação de tesouras.

As técnicas de alisamento utilizam somas ponderadas das jogadas anteriores. No caso específico do alisamento exponencial, pesos menores são dados para jogadas mais antigas, e pesos maiores, para jogadas mais recentes.

Ao executar o programa, ele inicialmente pedirá que você entre um "fator de esquecimento". Quanto maior o valor (inteiro, positivo) que você fornecer, maior o número de jogadas anteriores que o computador deixará de considerar. Assim, ficará mais fácil vencê-lo.

S

```
5 CLEAR 31999· GOSUB 500:
BORDER 0: PAPER 0: INK 7·
CLS
7 DIM A$(3,9): LET A$(1)="PE
DRA"· LET A$(2)="PAPEL"· LET
A$(3)="TESOURA"
10 DIM H(3,2)  DIM X(2)· DIM
Q(3,3)· DIM C(2,3): DIM A(2,3
): DIM P(3,3): CLS
20 FOR I=1 TO 3: LET H(I,1)=-
COS ((I 2)*PI*2/3): LET H(I,2
)=-SIN ((I-2)*PI*2/3): NEXT I
30 FOR T=1 TO 3: FOR J=1 TO 2
· LET A(J,T)=.01· NEXT J
NEXT T
40 INPUT "DIGITE FATOR DE ESQ
UECIMENTO     (0-1)  VALOR SUG
ERIDO  .85 ":W
70 FOR I=1 TO 3: FOR J=1 TO 3
· LET P(I,J)=SGN (I-J-3*INT (
(I-J+1.5)/3))· NEXT J: NEXT I
: LET S=0
80 LET U=1: LET S=0: LET U2=0
: LET WW=0: LET U3=0
100 LET V=INT (RND*3)+1
110 PRINT  INK 6; PAPER 2;"  1
=PEDRA, 2=PAPEL, 3=TESOURA  "
120 INK 6: PLOT 8,167  DRAW
239,0: DRAW 0,-159: DRAW -239,
0: DRAW 0,159
200 FOR T=1 TO 3: LET I=U
210 IF (U2=0 AND T=2) OR (U3=0
AND T=3) THEN  GOTO 280
220 IF T=2 THEN  LET I=ABS ((U
=U2)-2*(U=VV)-3*((U<>U2) AND (
U<>VV)))
230 IF T=3 THEN  LET I=ABS ((U
=U3)-2*(U=V3)-3*((U<>U3) AND (
U<>V3)))
240 FOR J=1 TO 2: LET A(J,T)=A
(J,T)*W+H(I,J): LET C(J,T)=C(J
,T)*W+3*H(I,J)*H(I,J)+.01
250 LET X(J)=A(J,T)/C(J,T):
NEXT J
260 FOR I=1 TO 3: LET Q(I,T)=1
/3: FOR K=1 TO 2: LET Q(I,T)=X
(K)*H(I,K): NEXT K: NEXT I
280 NEXT T: LET U2=U: LET VV=V
: IF U=V THEN  LET VV=U+1-3*
INT (U/3)
290 IF U<>V THEN  LET U3=U:
LET V3=V· IF P(U3,V3)<0 THEN
LET WW=U3: LET U3=V3: LET V3=
WW
300 LET X=-1E30: FOR T=1 TO 3:
IF (T=2 AND U2=0) OR (T=3 AND
U3=0) THEN  GOTO 370
310 IF T=1 THEN  GOTO 350
311 IF T=2 THEN  GOTO 320
312 IF T=3 THEN  GOTO 340
320 LET WW=6-U2-VV: LET Q1=Q(1
,T): LET Q2=Q(2,T): LET Q3=Q(3
,T): LET Q(U2,T)=Q1· LET Q(VV,
T)=Q2: LET Q(WW,T)=Q3
330 GOTO 350
340 LET WW=6-U3-V3: LET Q1=Q(1
,T): LET Q2=Q(2,T): LET Q3=Q(3
 T): LET Q(U3,T)=Q1· LET Q(V3,
T)=Q2: LET Q(WW,T)=Q3
350 FOR G=1 TO 3· LET P=0: FOR
I=1 TO 3: LET P P+P(G,I)*Q(I,T
): NEXT I: IF P>X THEN  LET X=
P: LET V=G
360 NEXT G
370 NEXT T
400 INK 7: PRINT AT 2,4;"VOCE
DISSE "
405 FOR M=-3 TO 16: SOUND .01,
M: NEXT M
410 LET K$=INKEY$: IF K$=""
THEN  GOTO 410
412 IF K$<"1" OR K$>"3" THEN
GOTO 410
415 LET U=VAL K$
420 PRINT AT 3,4;A$(U);AT 2,21
; INK 5;"EU DISSE ";AT 3,21;A$
(V)
430 POKE 23681,U-1: LET O=USR
32000· POKE 23681,127+V· LET O
=USR 32000
440 LET S=S+P(U,V): PRINT AT
16,8;"SEU SCORE E ";S;"  "
450 INVERSE 1: IF V=U THEN
PRINT AT 18,10;" UM EMPATE ":
GOTO 490
460 IF (U=3 AND V=2) OR (U=2
AND V=1) OR (U=1 AND V=3) THEN
 PRINT AT 18,12;" VOCE VENCEU
": GOTO 490
470 PRINT AT 18,11;" EU VENCI"
490 INVERSE 0: FOR D=1 TO 2:
PAUSE 0: NEXT D: FOR N=2 TO 18
: PRINT  PAPER 0; INK 7;AT N,2
;"
": NEXT N: GOTO 200
500 FOR N=32000 TO 32284: READ
A: PRINT N,A: NEXT N
505 LET N=32069: POKE 23728,N-
256*INT (N/256): POKE 23729,
INT (N/256)
510 RETURN
520 DATA 33,6,72,58,129,92,203
,127,40,5,203,191,33,22,72,221
,42,176,92,17,72,0,254,0,40
530 DATA 8,254,1,40,2,221,25,
221,25,221,229,209,6,3,197,229
,6,8,197,1,3,0,235,237,176,235
,36,1,3,0
540 DATA 237,66,193,16,239,225
,1,32,0,9,193,16,227,201
550 DATA 0,0,0,0,0,0,0,0,0,0,0
,0,0,30,0,0,97,128,3,129,192,4
,15,225,10,56,112,21,32,48,30,
192,16,11,160,16,7,120,32
560 DATA 3,181,96,1,234,192,0,
255,128,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0
570 DATA 0,16,0,0,56,0,0,124,0
,0,254,0,1,223,0,3,111,128,7,
191,192,14,219,224,27,109,240,
53,254,248,106,219,112,245,109
,224,122
580 DATA 190,192,61,91,128,30,
239,0,15,182,0,7,92,0,2,232,0,
1,240,0,0,224,0,0,0,0
590 DATA 0,0,0,0,0,0,0,0,0
600 DATA 0,0,0,0,0,0,0,0,56,0,
0,104,0,0,208,0,1,160,0,3,64,0
,6,128,0,13,0,0,26,0,0,52,0,1,
232,0,63,48,0
610 DATA 100,32,0,196,96,0,201
,192,0,115,96,0,6,32,0,4,32,0,
6,64,0,3,128,0,0,0,0,0,0,0,0,0
,0
```

T

```
5 CLEAR 1000:PMODE 3,1:COLOR 4,
2:PCLS
6 GOSUB 1000
7 C=8*ATN(1)/3
10 DIM H(3,2),X(2),Q(3,3),C(2,3
),A(2,3),P(3,3) CLS
20 FOR I=1 TO 3:H(I,1)=-COS((I-
2)*C):H(I,2)=-SIN((I-2)*C):NEXT
30 FOR T=1 TO 3:FOR J=1 TO 2:A(
J,T)=0:C(J,T)=.01·NEXT J,T
40 PRINT"DIGITE O FATOR DE ESQU
ECIMENTO. INTERVALO PERMITIDO
 0 A 1."
50 PRINT"0=SEM MEMORIA ALEM DE
1 JOGADA":PRINT"1=ESTRATEGIA FI
XA"
60 PRINT"VALOR SUGERIDO=.85":IN
PUT W
70 FOR I=1 TO 3:FOR J=1 TO 3:FO
R J=1 TO 3:P(I,J)=SGN(I-J-3*INT
((I-J+1 5)/3))·NEXT J,I
80 S=0·U2=0:WW=0:U3=0
100 V=RND(3)
110 COLOR 4:PCLS:LINE(8,20)-(24
```

```
7,171),PSET,B
120 DRAW"BM34,10S4C4" FOR K=1 T
O 3:DRAW N$(K)+E$+A$(K)+"BR4".N
EXT
140 GOTO 400
200 FOR T=1 TO 3 I=U
210 IF(U2=0 AND T=2) OR (U3=0 A
ND T=3) THEN 280
220 IF T=2 THEN I=-(U=U2)-2*(U=
VV)-3*((U<>U2)AND(U<>VV))
230 IF T=3 THEN I=-(U=U3)-2*(U=
V3)-3*((U<>U3)AND(U<>V3))
240 FOR J=1 TO 2:A(J,T)=A(J,T)*
W+H(I,J):C(J,T)=C(J,T)*W+3*H(I,
J)*H(I,J)+.01
250 X(J)=A(J,T)/C(J,T):NEXT
260 FOR I=1 TO 3 Q(I,T)=1/3 FOR
 K=1 TO 2 Q(I,T)=Q(I,T)+X(K)*H(
I,K) NEXT K,I
280 NEXT:U2=U.VV=V.IF U=V THEN
VV=U+1-3*INT(U/3)
290 IF U<>V THEN U3=U:V3=V:IF P
(U3,V3)<0 THEN WW=U3 U3=V3 V3=W
W
300 X=-1E30:FOR T=1 TO 3:IF(T=2
 AND U2=0) OR (T=3 AND U3=0) TH
EN 370
310 ON T GOTO 350,320,340
320 WW=6-U2-VV:Q1=Q(1,T):Q2=Q(2
,T):Q3=Q(3,T) Q(U2,T)=Q1 Q(VV,T
)=Q2 Q(WW,T)=Q3
330 GOTO 350
340 WW=6-U3-V3:Q1=Q(1,T):Q2=Q(2
,T):Q3=Q(3,T):Q(U3,T)=Q1:Q(V3,T
)=Q2 Q(WW,T)=Q3
350 FOR G=1 TO 3:P=0:FOR I=1 TO
 3:P=P+P(G,I)*Q(I,T):NEXT:IF P>
X THEN X=P:V=G
360 NEXT G
370 NEXT T
400 SCREEN 1,0:DRAW"BM12,30"+Y$
+S$
410 A$=INKEY$:IF A$<"1" OR A$>"
3" THEN 410
420 U=VAL(A$) DRAW A$(U)+"BM142
,30"+I$+S$+A$(V)
430 X=44 Y=90:ON U GOSUB 600,61
0,620
435 X=170 ON V GOSUB 700,710,72
0
440 LINE(80,175)-(240,190),PRES
ET,BF:S=S+P(U,V):DRAW"BM80,180B
D4R5U2L4U2R4BR8L4D4R4BR4U4R4D4N
L4BR4U4R4D2L2DFBR9L4U2NR4U2R4BR
8BU2S8":GOSUB 800:DRAW"S4"
450 IF U=V THEN DRAW"BM100,160C
3"+D$ GOTO 490
460 IF(U=3 AND V=2) OR (U2 AND
V=1) OR (U=1 AND V=3) THEN DRAW
"BM30,160C3"+Y$+W$ GOTO 490
470 DRAW"BM160,160C1"+I$+W$
490 FOR K=1 TO 1000:NEXT:LINE(1
0,22) (245,169),PRESET,BF
500 GOTO 200
600 PUT(X,Y)-(X+40,Y+38),PA,PSE
T RETURN
610 PUT(X,Y)-(X+40,Y+38),SC,PSE
T RETURN
620 PUT(X,Y)-(X+40,Y+38),ST,PSE
T RETURN
700 PUT(X,Y)-(X+40,Y+38),PA,AND
 RETURN
710 PUT(X,Y)-(X+40,Y+38),SC,AND
 RETURN
720 PUT(X,Y)-(X+40,Y+38),ST,AND
,RETURN
800 FOR K=1 TO LEN(STR$(S))
810 B$=MID$(STR$(S),K,1):IF B$=
"-" THEN DRAW"BF2R4BE2" GOTO 83
0
820 IF B$<"0" OR B$>"9" THEN 83
0
825 DRAW N$(VAL(B$))
830 NEXT RETURN
1000 FOR I=0 TO 9:READ N$(I):NE
XT
1010 DATA NR2D4R2U4BR2,BDEND4BR
2,R2D2L2D2R2BU4BR2,NR2BD2NR2BD2
R2U4BR2,D2R2D2U4BR2,NR2D2R2D2L2
BE4,D4R2U2L2BE2BR2,R2ND4BR2,NR2
D4R2U2NL2U2BR2,NR2D2R2D2U4BR2
1015 DIM PA(39),SC(39),ST(39)
1020 DRAW"BM0,22C3M22,0M40,18M2
0,38L4M0,22":PAINT(20,20)
1030 DRAW"BM32,20C2S8H4BH2HBG2F
4BFF3BL4H2BHH2BL3F2BF2F3BD3H2BH
3H2"
1040 GET(0,0)-(40,38),PA,G
1050 DRAW"BM50,30C3URURUR5S4UNR
6US8R3URNE8UE7R2G2DG6LD2LD2FDGL
GL2ULUEERE2L2DBM-3,-2D2G2L2U2"
1060 GET(50,0)-(90,38),SC,G
1070 DRAW"BM128,6L3GLGL2G3DF5R8
E2UEU3H3LU":PAINT(124,16)
1080 DRAW"BM128,8C2L3GLGNL2DFRU
ERER2U":PAINT(122,10):DRAW"BM13
4,14L3GLNG3D2RNF2R4NG2EUH":PAIN
T(130,18)
1090 GET(100,0)-(140,38),ST,G
1100 E$="BR2BDNR3BD2R3BE3BR"
1102 A$(1)="NR2D4U2R4U2BF4U2NR2
U2R4D4BR4U2NU2R5U2NL2BR8L4D2NR2
D2R4BR3U4R4D2LDFRBE4"
1104 A$(2)="BD4R4U2L3U2R4BR8L4D
4R4BR4U4BR8L4D2R3D2L3BR7R4U2L3U
2R4BR3ND4R5D4NL2BR3U4R4D2LDFBR5
R3U2L3U2R4BR2"
1110 A$(3)="BD4R4U2L3U2R4BR3R3N
D4R2BR3NR5D4R5U4BR4ND4F3RFU4BF4
NR4U2NR2U2R5BR3"
1120 I$="ND4BR6":Y$="C4F2ND2RE2
BR3D4R4U4LBR6D4R3U4BR6"
1130 W$="D4RERERFRFU4BR4ND4BR5N
D4F2RF2U4"
1140 S$="BD4R4U2L3U2R4BR3ND4R4D
2NLD2BR4U4BR4D4R3EU2HBR9"
1150 D$="ND4R4D2NLD2BR7U4R2FD2G
BR5U4R4D2LDFBR4U4R4D2LF2BR3NU4E
RERFRFU4"
1200 RETURN
```

Nas versões para o MSX, o Apple e o TK-2000 não há apresentação de gráficos.



```
6 GOSUB 1000
7 C=8*ATN(1)/3:NU=0:NV=0
10 DIM H(3,2),X(2),Q(3,3),C(2,3
),A(2,3),P(3,3)
20 FOR I=1 TO 3
25 H(I,1)=-COS((I-2)*C):H(I,2)
=-SIN((I-2)*C)
27 NEXT I
30 FOR T=1 TO 3:FOR J=1 TO 2:A(
J,T)=0:C(J,T)=0.01:NEXT J:NEXT
T
35 LOCATE 0,4
40 PRINT "FATOR DE ESQUECIMENTO
 A SER USADO"
45 PRINT:PRINT "VALOR ENTRE 0 E
 1 :"
50 PRINT "0 = MEMORIA DE 1 JOGO
 APENAS":PRINT "1 = ESTRATEGIA
INVARIAVEL"
60 PRINT "VALOR SUGERIDO = 0.85
"
65 PRINT:PRINT "VALOR ";:INPUT
W
67 FOR I=4 TO 11:LOCATE 0,I
68 PRINT "
                    "
69 NEXT I
70 FOR I=1 TO 3 FOR J=1 TO 3
72 P(I,J)=1
75 IF I-J-3*INT((I-J+1.5)/3)<0
THEN P(I,J)=-1
77 NEXT J NEXT I
80 S=0 U2=0 WW=0 U3=0
100 V=INT(3*RND(1))+1
120 LOCATE 3,4:PRINT "VOCE"
130 LOCATE 20,4:PRINT "EU"
140 GOTO 400
200 FOR T=1 TO 3:I=U
210 IF (U2=0 AND T=2) OR (U3=0
AND T=3) THEN 270
220 IF T=2 THEN I=-(U=U2)-2*(U=
VV)-3*((U<>U2) AND (U<>VV))
230 IF T=3 THEN I=-(U=U3)-2*(U=
V3)-3*((U<>U3) AND (U<>V3))
240 FOR J=1 TO 2:LET A(J,T)=A(J
,T)*W+H(I,J)
245 C(J,T)=C(J,T)*W+3*H(I,J)*
H(I,J)+0.01
250 LET X(J)=A(J,T)/C(J,T):NEXT
 J
260 FOR I=1 TO 3:Q(I,T)=1/3:FOR
 K=1 TO 2
265 Q(I,T)=Q(I,T)+X(K)*H(I,K):
NEXT K NEXT I
270 NEXT T
280 U2=U VV=V
285 IF U=V THEN VV=U+1-3*INT(U/
3)
290 IF U<>V THEN U3=U:V3=V IF P
(U3,V3)<0 THEN WW=U3:U3=V3 V3=W
W
300 X=-9999999:FOR T=1 TO 3
305 IF (T=2 AND U2=0) OR (T=3 A
ND U3=0) THEN 370
310 ON T GOTO 350,320,340
320 WW=6-U2-VV:Q1=Q(1,T) Q2=
Q(2,T):Q3=Q(3,T)
325 Q(U2,T)=Q1 Q(VV,T)=Q2:Q(WW
,T)=Q3
330 GOTO 350
340 WW=6-U3-V3:Q1=Q(1,T) Q2=Q(
2,T):Q3=Q(3,T)
345 Q(U3,T)=Q1:Q(V3,T)=Q2:Q(WW
,T)=Q3
350 FOR G=1 TO 3:P=0:FOR I=1 TO
 3
355 LET P=P+P(G,I)*Q(I,T):NEXT
I
357 IF P>X THEN X=P V=G
360 NEXT G
370 NEXT T
```

```
400 LOCATE 0,21:PRINT "JOGUE: 1
=PAPEL 2=TESOURA 3=PEDRA ";
410 A$=INKEY$:IF A$<"1" OR A$>"
3" THEN 410
420 U=VAL(A$)
425 LOCATE 3,10:PRINT A$(U)
430 LOCATE 20,10:PRINT A$(V)
445 S=S+P(U,V)
447 LOCATE 12,16:PRINT "
       ":LOCATE 12,16
450 IF U=V THEN PRINT "EMPATE":
GOTO 490
460 IF (U=3 AND V=2) OR (U=2 AN
D V=1) OR (U=1 AND V=3) THEN PR
INT "VOCE GANHOU !":NU=NU+1:GOT
O 490
470 PRINT "GANHEI !":NV=NV+1
490 LOCATE 3,18:PRINT NU:LOCATE
 20,18:PRINT NV
500 GOTO 200
1000 LET A$(1)="PAPEL   ":A$(2)=
"TESOURA":A$(3)="PEDRA   "
1010 CLS
1020 PRINT "P A P E L  -  P E D
 R A - T E S O U R A"
1030 PRINT "==================
==================="
1040 LOCATE 0,20
1050 PRINT "========= =========
====================="
1200 RETURN
```



```
6 GOSUB 1000
7 LET C=8*ATN(1)/3:NU=0:NV=0
10 DIM H(3,2),X(2),Q(3,3),C(2,3
   ),A(2,3),P(3,3)
20 FOR I=1 TO 3
25 LET H(I,1)=-COS((I-2)*C):H(I
   ,2)=-SIN((I-2)*C)
27 NEXT I
30 FOR T=1 TO 3:FOR J=1 TO 2:A(
   J,T)=0:C(J,T)=0.01:NEXT J:
   NEXT T
35 VTAB 4:HTAB 1
40 PRINT "FATOR DE ESQUECIMENTO
   A SER USADO"
45 PRINT:PRINT "VALOR ENTRE 0 E
   1 :"
50 PRINT "0 = MEMORIA DE 1 JOGO
   APENAS":PRINT "1 = ESTRATEGI
   A INVARIAVEL"
60 PRINT "VALOR SUGERIDO = 0.85
   "
65 PRINT:PRINT "VALOR ";:INPUT
   W
67 FOR I=4 TO 11:VTAB I:HTAB 1
68 PRINT "
                     "
69 NEXT I
70 FOR I=1 TO 3:FOR J=1 TO 3
72 LET P(I,J)=1
75 IF I-J-3*INT((I-J+1.5)/3)<0
   THEN P(I,J)=-1
77 NEXT J:NEXT I
80 LET S=0:U2=0:WW=0:U3=0
100 LET V=INT(3*RND(1))+1
120 VTAB 4:HTAB 3:PRINT "VOCE"
130 VTAB 4:HTAB 20:PRINT "EU"
140 GOTO 400
200 FOR T=1 TO 3:I=U
210 IF (U2=0 AND T=2) OR (U3=0
    AND T=3) THEN 270
220 IF T=2 THEN I=-(U=U2)-2*(U=
    VV)-3*((U<>U2) AND (U<>VV))
230 IF T=3 THEN I=-(U=U3)-2*(U=
    V3)-3*((U<>U3) AND (U<>V3))
240 FOR J=1 TO 2:LET A(J,T)=A(J
    ,T)*W+H(I,J)
245 LET C(J,T)=C(J,T)*W+3*H(I,J
    )*H(I,J)+0.01
250 LET X(J)=A(J,T)/C(J,T):NEXT
    J
260 FOR I=1 TO 3:Q(I,T)=1/3:FOR
    K=1 TO 2
265 LET Q(I,T)=Q(I,T)+X(K)*H(I,
    K):NEXT K:NEXT I
270 NEXT T
280 LET U2=U:VV=V
285 IF U=V THEN VV=U+1-3*INT(U/
    3)
290 IF U<>V THEN U3=U:V3=V:IF P
    (U3,V3)<0 THEN WW=U3 : U3 =
    V3:V3=WW
300 LET X=-9999999:FOR T=1 TO 3
305 IF (T=2 AND U2=0) OR (T=3 A
    ND U3=0) THEN 370
310 ON T GOTO 350,320,340
320 LET WW=6-U2-VV:Q1=Q(1,T):Q2
    =Q(2,T):Q3=Q(3,T)
325 LET Q(U2,T)=Q1:Q(VV,T)=Q2:Q
    (WW,T)=Q3
330 GOTO 350
340 LET WW=6-U3-V3:Q1=Q(1,T):Q2
    =Q(2,T):Q3=Q(3,T)
345 LET Q(U3,T)=Q1:Q(V3,T)=Q2:Q
    (WW,T)=Q3
350 FOR G=1 TO 3:P=0:FOR I=1 TO
    3
355 LET P=P+P(G,I)*Q(I,T):NEXT
    I
357 IF P>X THEN X=P:V=G
360 NEXT G
370 NEXT T
400 VTAB 21:HTAB 1:PRINT"JOGUE:
    1=PAPEL 2=TESOURA 3=PEDRA";
410 GET A$:IF A$<"1" OR A$>"3"
    THEN 410
420 LET U=VAL(A$)
425 VTAB 10:HTAB 3:PRINT A$(U)
430 VTAB 10:HTAB 20:PRINT A$(V)
445 LET S=S+P(U,V)
447 VTAB 16:HTAB 12:PRINT "
         ":VTAB 16:HTAB 12
450 IF U=V THEN PRINT "EMPATE":
    GOTO 490
460 IF (U=3 AND V=2) OR (U=2 AN
    D V=1) OR (U=1 AND V=3)THEN
    PRINT "VOCE GANHOU !":NU=NU
    +1:GOTO 490
470 PRINT "GANHEI !":NV=NV+1
490 VTAB 18:HTAB 3:PRINT NU
    VTAB 18:HTAB 20:PRINT NV
500 GOTO 200
1000 LET A$(1)="PAPEL   ":A$(2)=
     "TESOURA":A$(3)="PEDRA   "
1010 HOME
1020 PRINT "P A P E L  -  P E D
     R A - T E S O U R A"
1030 PRINT "==================
     ==================="
1040 VTAB 20:HTAB 1
1050 PRINT "==================
     ==================="
1200 RETURN
```

SOMA CUMULATIVA

Soma cumulativa é a técnica usada para analisar o desempenho do oponente num período de tempo em que se registram mudanças bruscas de respostas.

Os gráficos empregados pelo programa são iguais aos do programa anterior (apenas nas versões para o TRS-Color e Spectrum. Nas versões para o MSX, Apple e TK-2000, a parte comum entre os dois programas se refere à apresentação dos resultados). Assim, as linhas de programa correspondentes à seção relativa à apresentação dos resultados não mudam, devendo ser incorporadas ao novo programa como estão.

Como no programa anterior, o jogador deve entrar uma informação que regula a tática do computador (um número entre 1 e 9999).



Use as linhas 520 a 610 do programa anterior e acrescente estas:

```
 5 CLEAR 31999: GOSUB 500:
BORDER 0: PAPER 0: INK 7:
CLS
 7 DIM A$(3,9): LET A$(1)="PE
DRA": LET A$(2)="PAPEL": LET
A$(3)="TESOURA"
10 DIM B(3): DIM U(3): DIM S(
3): DIM H(3,2): DIM X(3): DIM
Q(3): DIM P(3,3): DIM M(3)
15 LET MM=60: DIM A(3,MM):
DIM Z(MM)
20 FOR I=1 TO 3: LET H(I,1)=-
COS ((I-2)*PI*2/3): LET H(I,2
)= SIN ((I-2)*PI*2/3): NEXT I
30 LET M(1)=1: LET M(2)=0:
LET M(3)=0
40 INPUT "DIGITE RAZAO DE SEM
ELHANCA PARA CADA JOGADA (1 A
 9999), 1=NOVA  ESTRATEGIA PA
RA CADA JOGADA,      9999 MESMA
 ESTRATEGIA. ",W
70 FOR I=1 TO 3: FOR J=1 TO 3
: LET P(I,J)=SGN (I-J-3*INT (
(I-J+1.5)/3)): NEXT J: NEXT I
: LET S=0
80 LET NN=0: LET U=1: LET S=0
: LET U2=0: LET WW=0: LET U3=0
100 LET V=INT (RND*3)+1
110 PRINT  INK 6; PAPER 2;"  1
=PEDRA, 2=PAPEL, 3=TESOURA  "
120 INK 6: PLOT 8,167: DRAW
239,0: DRAW 0,-159: DRAW -239,
0: DRAW 0,159
130 GOTO 400
200 LET Y=-1E30: LET Z=-1E30:
FOR T=1 TO 3: LET I=U: IF T=1
THEN  GOTO 230
210 IF (U2=0 AND T=2) OR (U3 0
AND T=3) THEN  GOTO 264
220 IF T=2 THEN  LET I=ABS ((U
=U2)-2*(U=VV)-3*((U<>U2) AND (
U<>VV)))
```



```
230 IF T=3 THEN  LET I=ABS ((U
=U3)-2*(U=V3)-3*((U<>U3) AND (
U<>V3)))
240 LET A(T,M(T))=I
250 FOR J=1 TO 3: LET B(J)=0:
NEXT J: LET N=0: LET N2=M(1)
252 FOR M=M(T) TO 1 STEP -1:
LET B(A(T,M))=B(A(T,M))+1: LET
N=N+1: LET N2=N2-1
256 FOR J=1 TO 3: LET Q(J)=B(J
)/N: LET X(J)=Q(J)+(Q(J)=0):
NEXT J
258 LET Q=B(1)*LN (X(1))+B(2)*
LN (X(2))+B(3)*LN (X(3)): IF
NN>1 AND N2<>0 THEN  LET Q=Q+Z
(N2)
260 IF Q>Z THEN  LET Z=Q: LET
S(1)=Q(1): LET S(2)=Q(2): LET
S(3)=Q(3): LET SS=T: LET NN=N2
262 NEXT M: IF Q>Y THEN  LET Y
=Q: LET TT=T: LET U(1)=Q(1)
LET U(1)=Q(2): LET U(3)=Q(3)
264 NEXT T: LET T=TT: LET Z(M(
1))=Y: LET Q(1)=U(1): LET Q(2)
=U(2): LET Q(3)=U(3)
270 LET U2=U: LET VV=V: IF U2=
VV THEN  LET VV=U2+1-3*INT (U2
/3)
272 IF U<>V THEN  LET U3=U
LET V3=V: IF P(U3,V3)<0 THEN
LET WW=U3: LET U3=V3: LET V3=
WW
274 LET M(1)=M(1)+1: LET M(2)=
M(2)+(U2>0): LET M(3)=M(3)=(U3
>0)
280 IF Q>Z-LN (W) THEN  GOTO
300
282 FOR T=1 TO 3: FOR M=NN+1
TO M(T): LET A(T,M-NN)=A(T,M):
NEXT M
284 LET M(T)=(NN M(T))*(-1*((M
(T)>NN))): NEXT T
286 LET T=TT: LET Q(1)=S(1):
LET Q(2)=S(2): LET Q(3)=S(3)
290 FOR M=1 TO M(1)-1: LET Z(M
)=-1E30: NEXT M  LET Z(M(1))=Q
300 LET X=-1E30
310 IF T=1 THEN  GOTO 350
311 IF T=2 THEN  GOTO 320
312 IF T=3 THEN  GOTO 340
320 LET WW=6-U2-VV: LET Q1=Q(1
): LET Q2=Q(2): LET Q3=Q(3):
LET Q(U2)=Q1: LET Q(VV)=Q2.
LET Q(WW)=Q3
330 GOTO 350
340 LET WW=6-U3-V3: LET Q1=Q(1
): LET Q2=Q(2): LET Q3=Q(3):
LET Q(U3)=Q1: LET Q(V3)=Q2:
LET Q(WW)=Q3
350 FOR G=1 TO 3: LET P=0: FOR
I=1 TO 3: LET P=P+P(G,I)*Q(I):
NEXT I: IF P>X THEN  LET X=P:
LET V=G
360 NEXT G
400 INK 7: PRINT AT 2,4;"VOCE
DISSE "
405 FOR M=-3 TO 16: SOUND .01,
M: NEXT M
410 LET K$=INKEY$: IF K$=""
THEN  GOTO 410
412 IF K$<"1" OR K$>"3" THEN
GOTO 410
415 LET U=VAL K$
420 PRINT AT 3,4;A$(U);AT 2,21
; INK 5;"EU DISSE ";AT 3,21;A$
(V)
430 POKE 23681,U-1: LET O=USR
32000: POKE 23681,127+V: LET O
=USR 32000
440 LET S=S+P(U,V): PRINT AT
16,8;"SEU PLACAR E ";S;"  "
450 INVERSE 1: IF V=U THEN
PRINT AT 18,10;" UM EMPATE ":
GOTO 490
460 IF (U=3 AND V=2) OR (U=2
AND V=1) OR (U=1 AND V=3) THEN
 PRINT AT 18,10;" VOCE VENCEU
": GOTO 490
470 PRINT AT 18,10;"  EU VENCI
 "
490 INVERSE 0: FOR D=1 TO 2:
PAUSE 0: NEXT D: FOR N=2 TO 18
: PRINT  PAPER 0; INK 7;AT N,2
;"                              "
: NEXT N: GOTO 200
500 FOR N=32000 TO 32284: READ
A: POKE N,A: NEXT N
505 LET N=32069: POKE 23728,N-
256*INT (N/256): POKE 23729,
INT (N/256)
510 RETURN
```



Apague o programa anterior até a linha 435 e digite:

```
5 CLEAR 1000:PMODE 3,1:COLOR 4,
2:PCLS
6 GOSUB 1000
7 C=8*ATN(1)/3
10 DIM B(3),U(3),S(3),H(3,2),X(
3),Q(3),P(3,3),M(3)
15 MM=60:DIM A(3,MM),Z(MM)
20 FOR I=1 TO 3:H(I,1)=-COS((I-
2)*C):H(I,2)=-SIN((I-2)*C):NEXT
30 M(1)=1
40 CLS:INPUT"DIGITE RAZAO DE SE
MELHANCA PARA CADA JOGADA (1 A
9999),  1=NOVA ESTRATEGIA PARA
CADA JOGADA ,    9999=MESMA ESTR
ATEGIA. ";W
70 FOR I=1 TO 3:FOR J=1 TO 3:P(
I,J)=SGN(I-J-3*INT((I-J+1.5)/3)
):NEXT J,I
80 U=1
100 V=RND(3)
110 COLOR 4:PCLS:LINE(8,20)-(24
7,171),PSET,B
120 DRAW"BM34,10S4C4":FOR K=1 T
O 3:DRAW N$(K)+E$+A$(K)+"BR4":N
EXT
140 GOTO 400
200 Y=-1E30:Z=-1E30:FOR T=1 TO
3:I=U:IF T=1 THEN 230
210 IF(U2=0 AND T=2)OR(U3=0 AND
 T=3) THEN 264
220 IF T=2 THEN I=-(U=U2)-2*(U=
VV)-3*((U<>U2)AND(U<>VV))
230 IF T=3 THEN I=-(U=U3)-2*(U=
V3)-3*((U<>U3)AND(U<>V3))
240 A(T,M(T))=I
250 FOR J=1 TO 3:B(J)=0:NEXT:N=
0:N2=M(1)
252 FOR M=M(T) TO 1 STEP -1:B(A
(T,M))=B(A(T,M))+1:N=N+1:N2=N2-
1
256 FOR J=1 TO 3:Q(J)=B(J)/N:X(
J)=Q(J)-(Q(J)=0):NEXT
258 Q=B(1)*LOG(X(1))+B(2)*LOG(X
(2))+B(3)*LOG(X(3)).IF NN>0 THE
N Q=Q+Z(N2)
260 IF Q>Z THEN Z=Q:S(1)=Q(1):S
(2)=Q(2):S(3)=Q(3):SS=T:NN=N2
262 NEXT M:IF Q>Y THEN Y=Q:TT=T
:U(1)=Q(1):U(2)=Q(2):U(3)=Q(3)
264 NEXT T:T=TT:Z(M(1))=Y:Q(1)=
U(1):Q(2)=U(2):Q(3)=U(3)
270 U2=U:VV=V:IF U2>VV THEN VV=
U2+1-3*INT(U2/3)
272 IF U<>V THEN U3=U:V3=V:IF P
(U3,V3)<0 THEN WW=U3:U3=V3:V3=W
W
274 M(1)=M(1)+1:M(2)=M(2)=(U2>0
):M(3)=M(3)-(U3>0)
280 IF Q>Z-LOG(W) THEN 300
282 FOR T=1 TO 3:FOR M=NN+1 TO
M(T):A(T,M-NN)=A(T,M).NEXT
284 M(T)=(NN-MT)*(M(T)>NN):NEXT
286 T=TT:Q(1)=S(1):Q(2)=S(2):Q(
3)=S(3)
290 FOR M=1 TO M(1)-1:Z(M)=-1E3
0:NEXT:Z(M(1))=Q
300 X=-1E30
310 ON T GOTO 350,320,340
320 WW=6-U2-VV:Q1=Q(1):Q2=Q(2):
Q3=Q(3):Q(U2)=Q1:Q(VV)=Q2:Q(WW)
=Q3
330 GOTO 350
340 WW=6-U3-V3:Q1=Q(1):Q2=Q(2):
Q3=Q(3):Q(U3)=Q1:Q(V3)=Q2:Q(WW)
=Q3
350 FOR G=1 TO 3:P=0:FOR I=1 TO
 3:P=P+P(G,I)*Q(I):NEXT:IF P>X
THEN X=P:V=G
360 NEXT
400 SCREEN 1,0:DRAW"BM12,30"+Y$
+S$
410 A$=INKEY$:IF A$<"1" OR A$>"
3" THEN 410
420 U=VAL(A$):DRAW A$(U)+"BM142
,30"+I$+S$+A$(V)
430 X=44:Y=90:ON U GOSUB 600,61
0,620
435 X=170:ON V GOSUB 700,710,72
0
```



Apague o programa anterior até a linha 390 e digite:

```
6 GOSUB 1000
7 C=8*ATN(1)/3:NU=0:NV=0
10 DIM B(3),U(3),S(3),H(3,2),X(
3),Q(3),P(3,3),M(3)
15 MM=60:DIM A(3,MM),Z(MM)
20 FOR I=1 TO 3
25 H(I,1)=-COS((I-2)*C):H(I,2)
=-SIN((I-2)*C)
27 NEXT I
30 M(1)=1
35 LOCATE 0,4
40 PRINT "FATOR DE VEROSSIMILHA
NÇA"
45 PRINT:PRINT "VALOR ENTRE 1 E
```

```
 9999 :"
50 PRINT "1 = NOVA ESTRATEGIA P
ARA CADA JOGO":PRINT "9999 = ME
SMA ESTRATEGIA"
65 PRINT:PRINT "VALOR ";:INPUT
W
67 FOR I=4 TO 10:LOCATE 0,I
68 PRINT "
                    "
69 NEXT I
70 FOR I=1 TO 3:FOR J=1 TO 3
72 P(I,J)=1
75 IF I-J-3*INT((I J+1.5)/3)<0
THEN P(I,J)=-1
77 NEXT J:NEXT I
80 U=1
100 V=INT(3*RND(1))+1
120 LOCATE 3,4:PRINT "VOCE"
130 LOCATE 20,4:PRINT "EU"
140 GOTO 400
200 Y=-99999:FOR T=1 TO 3:I=U
205 IF T=1 THEN 230
210 IF (U2=0 AND T=2) OR (U3=0
AND T=3) THEN 264
220 IF T=2 THEN I=-(U=U2)-2*(U=
VV)-3*((U<>U2) AND (U<>VV))
230 IF T=3 THEN I=-(U=U3)-2*(U=
V3)-3*((U<>U3) AND (U<>V3))
240 A(T,M(T))=I
250 FOR J=1 TO 3:B(J)=0:NEXT J:
N=0:N2=M(1)
252 FOR M=M(T) TO 1 STEP -1
254 B(A(T,M))=B(A(T,M))+1:N=N+1
:N2=N2-1
256 FOR J=1 TO 3:Q(J)=B(J)/N:X(
J)=Q(J)-(Q(J)=0):NEXT J
258 A=B(1)*LOG(X(1))+B(2)*LOG(X
(2))+B(3)*LOG(X(3))
259 IF NN>0 THEN Q=Q+Z(N2)
260 IF Q>Z THEN Z=Q:S(1)=Q(1):S
(2)=Q2:S(3)=Q3:SS=T:NN=N2
262 NEXT M
263 IF Q>Y THEN Y=Q:TT=T:U(1)=Q
(1):U(2)=Q(2):U(3)=Q(3)
264 NEXT T
265 T=TT:Z(M(1))=Y:Q(1)=U(1):
Q(2)=U(2):Q(3)=U(3)
270 U2=U:VV=V:IF U2>VV THEN
 VV=U2+1-3*INT(U2/3)
272 IF U<>V THEN U3=U:V3=V:IF P
(U3,V3)<0 THEN WW=U3:U3=V3:V3=W
W
274 M(1)=M(1)+1:LET M(2)=M(2)-(
U2>0):M(3)=M(3)-(U3>0)
280 IF Q>Z-LOG(W) THEN 300
282 FOR T=1 TO 3:FOR M=NN+1 TO
M(T)
283 A(T,M-NN)=A(T,M):NEXT M
284 M(T)=(NN-MT)*(M(T)>NN):
NEXT T
286 T=TT:Q(1)=S(1):Q(2)=S(2):
Q(3)=S(3)
290 FOR M=1 TO M(1)-1:Z(M)=
-99999:NEXT M
295 Z(M(1))=Q
300 X=-99999
310 ON T GOTO 350,320,340
320 WW=6-U2-VV:Q1=Q(1):Q2=Q(2):
Q3=Q(3)
325 Q(U2)=Q1:Q(VV)=Q2:Q(WW)=Q3
330 GOTO 350
340 WW=6-U3-V3:Q1=Q(1):Q2=Q(2):
Q3=Q(3)
345 Q(U3)=Q1:Q(V3)=Q2:Q(WW)=Q3
350 FOR G=1 TO 3:P=0:FOR I=1 TO
 3
355 P=P+P(G,I)*Q(I):NEXT I
357 IF P>X THEN X=P:V=G
360 NEXT G
```



Apague o programa anterior até a linha 390 e digite:

```
6 GOSUB 1000
7 LET C=8*ATN(1)/3:NU=0:NV=0
10 DIM B(3),U(3),S(3),H(3,2),X(
   3),Q(3),P(3,3),M(3)
15 LET MM=60:DIM A(3,MM),Z(MM)
20 FOR I=1 TO 3
25 LET H(I,1)=-COS((I-2)*C):H(I
   ,2)=-SIN((I-2)*C)
27 NEXT I
30 LET M(1)=1
35 VTAB 4:HTAB 1
40 PRINT "FATOR DE VEROSSIMILHA
   NCA"
45 PRINT:PRINT "VALOR ENTRE 1 E
    9999 :"
50 PRINT "1 = NOVA ESTRATEGIA P
   ARA CADA JOGO":PRINT "9999 =
    MESMA ESTRATEGIA"
65 PRINT:PRINT "VALOR ";:INPUT
   W
67 FOR I=4 TO 10:VTAB I:HTAB 1
68 PRINT "
                    "
69 NEXT I
70 FOR I=1 TO 3:FOR J=1 TO 3
72 LET P(I,J)=1
75 IF I-J-3*INT((I-J+1.5)/3)<0
   THEN P(I,J)=-1
77 NEXT J:NEXT I
80 LET U=1
100 LET V=INT(3*RND(1))+1
120 VTAB 4:HTAB 3:PRINT "VOCE"
130 VTAB 4:HTAB 20:PRINT "EU"
140 GOTO 400
200 LET Y=-99999:FOR T=1 TO 3:I
    =U
205 IF T=1 THEN 230
210 IF (U2=0 AND T=2) OR (U3=0
    AND T=3) THEN 264
220 IF T=2 THEN I=-(U=U2)-2*(U=
    VV)-3*((U<>U2) AND (U<>VV))
230 IF T=3 THEN I=-(U=U3)-2*(U=
    V3)-3*((U<>U3) AND (U<>V3))
240 LET A(T,M(T))=I
250 FOR J=1 TO 3:B(J)=0:NEXT J:
    N=0:N2=M(1)
252 FOR M=M(T) TO 1 STEP -1
254 LET B(A(T,M))=B(A(T,M))+1:N
    =N+1:N2=N2-1
256 FOR J=1 TO 3:Q(J)=B(J)/N:X(
    J)=Q(J)-(Q(J)=0):NEXT J
258 LET A=B(1)*LOG(X(1))+B(2)*L
    OG(X(2))+B(3)*LOG(X(3))
259 IF NN>0 THEN Q=Q+Z(N2)
260 IF Q>2 THEN 2=Q:S(1)=Q(1):S
    (2)=Q2:S(3)=Q3:SS=T:NN=N2
262 NEXT M
263 IF Q>Y THEN Y=Q:TT=T:U(1)=Q
    (1):U(2)=Q(2):U(3)=Q(3)
264 NEXT T
```

MICRO DICAS

COMO ENGANAR O COMPUTADOR

Se você não quiser ser derrotado pelo computador, precisará lançar mão de uma estratégia que neutralize ao máximo o método de análise utilizado no programa. A melhor alternativa, nesse caso, é empregar um processo absolutamente aleatório em suas jogadas. Para isso, rode um pequeno programa de sorteio de três números no seu micro, e imprima (ou então copie à mão) uma longa sequência de jogadas. Seguindo-a à risca, o computador não terá nenhuma chance de prever suas jogadas. Em compensação, você também não conseguirá prever as jogadas do computador!

Outra técnica para enganar o computador consiste em mudar continuamente de estratégia. Os resultados, aqui, dependerão do programa que você estiver usando. Veja como o computador "enlouquece"...

```
265 LET T=TT:Z(M(1))=Y:Q(1)=U(1
    ):Q(2)=U(2):Q(3)=U(3)
270 LET U2=U:VV=V:IF U2>VV THEN
     VV=U2+1-3*INT(U2/3)
272 IF U<>V THEN U3=U:V3=V:IF P
    (U3,V3)<0 THEN WW=U3:U3=V3:
    V3=WW
274 LET M(1)=M(1)+1:LET M(2)=M(
    2)-(U2>0):M(3)=M(3)-(U3>0)
280 IF Q>Z-LOG(W) THEN 300
282 FOR T=1 TO 3:FOR M=NN+1 TO
    M(T)
283 LET A(T,M-NN)=A(T,M):NEXT M
284 LET M(T)=(NN-MT)*(M(T)>NN):
    NEXT T
286 LET T=TT:Q(1)=S(1):Q(2)=S(2
    ):Q(3)=S(3)
290 FOR M=1 TO M(1)-1:LET Z(M)=
    -99999:NEXT M
295 LET Z(M(1))=Q
300 LET X=-99999
310 ON T GOTO 350,320,340
320 LET WW=6-U2-VV:Q1=Q(1):Q2=Q
    (2):Q3=Q(3)
325 LET Q(U2)=Q1:Q(VV)=Q2:Q(WW)
    =Q3
330 GOTO 350
340 LET WW=6-U3-V3:Q1=Q(1):Q2=Q
    (2):Q3=Q(3)
345 LET Q(U3)=Q1:Q(V3)=Q2:Q(WW)
    =Q3
350 FOR G=1 TO 3:P=0:FOR I=1 TO
    3
355 LET P=P+P(G,I)*Q(I):NEXT I
357 IF P>X THEN X=P:V=G
360 NEXT G
```

A MATEMÁTICA DA IRREGULARIDADE (1)

Use seu micro na exploração das formas mais fascinantes da geometria de dimensões fracionadas, um instrumento da matemática que ajuda a explicar as irregularidades observadas na natureza.

Qual é o comprimento de um pedaço de corda? A resposta é muito fácil se a corda estiver esticada, pois bastará medi-la com uma fita métrica. Caso contrário, a tarefa será um pouco mais trabalhosa, pois teremos que medir um objeto irregular.

É evidente que, para obter as medidas de um objeto irregular, procuraremos estendê-lo, de modo a simplificar o trabalho. Porém, nem todas as coisas são suficientemente maleáveis para se fazer isso. Como agir então?

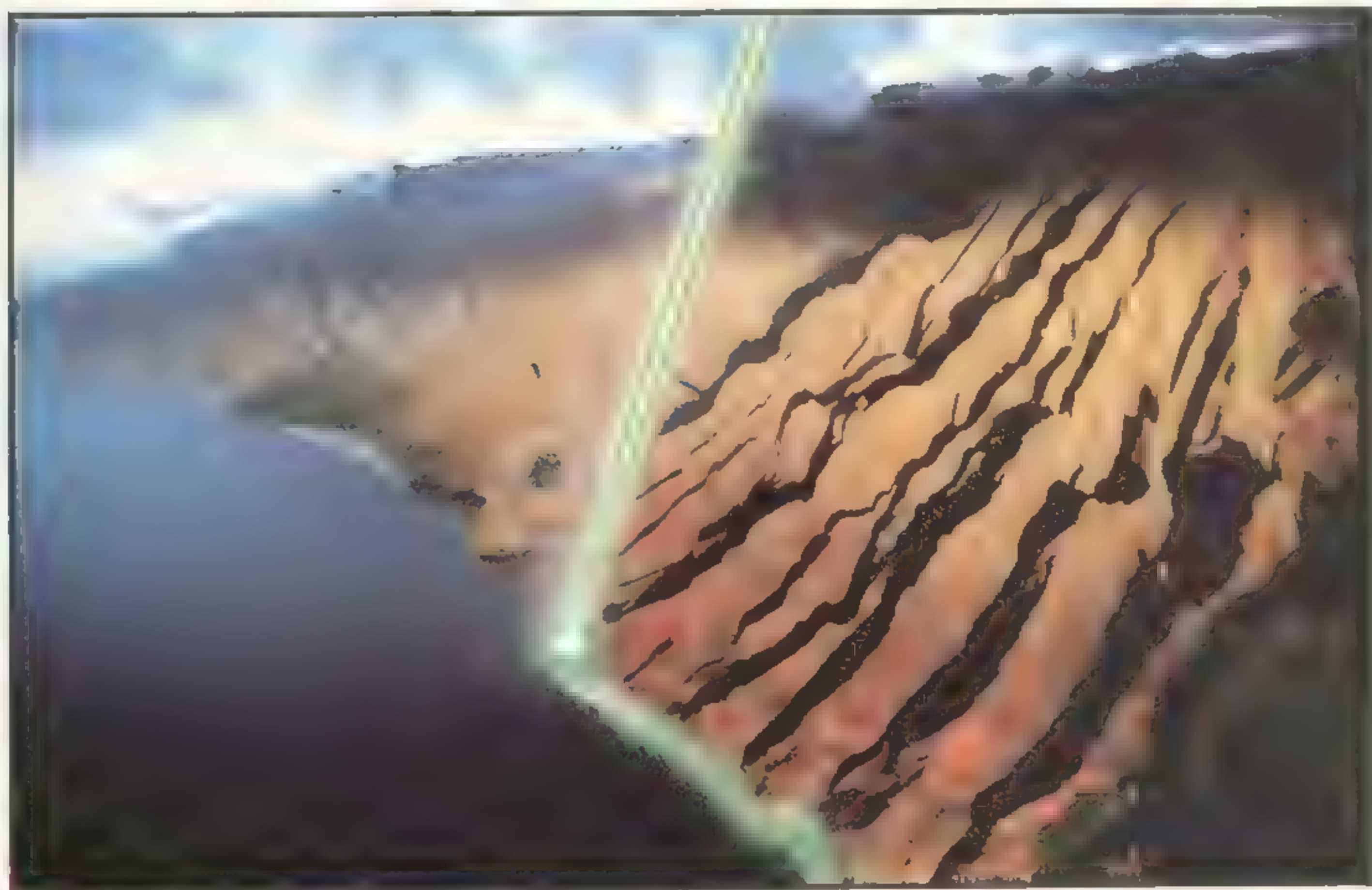
Tomemos como exemplo uma encosta rochosa com 5 km de extensão. Essa medida equivale ao percurso de uma gaivota que voa de uma ponta à outra ou à distância que você realmente andaria seguindo a linha do mar? Há uma diferença considerável entre os dois trajetos, sendo que a rota que acompanha todas as irregularidades é bem maior que a do vôo retilíneo do pássaro.

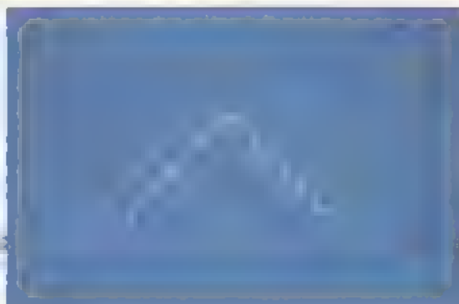
Mas qual é essa diferença? Suponhamos que temos uma trena bem flexível, capaz de se ajustar a todos os pormenores do terreno. Inicialmente, vamos nos ater aos grandes acidentes, como a foz de um rio ou uma baía. Mas, em busca de maior rigor, passaremos a considerar cada rocha que se projeta ao mar, tornando a costa irregular, e, depois, cada pedra à beira d'água. Por fim, levaremos em conta até os grãos de areia: se você os observar em detalhe, verá que também são irregulares e merecem uma boa inspeção.

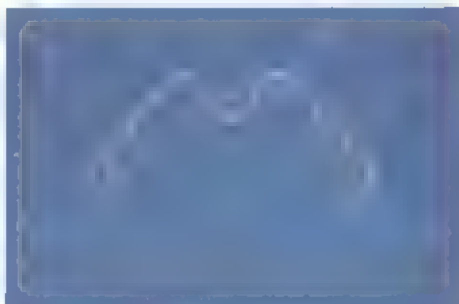
Isso parece um exercício absurdo — afinal, ninguém precisa de uma medida desse tipo com tamanha precisão. Mas, por meio do exemplo dado, pudemos constatar algo importante: a exatidão da medida de uma superfície irregular depende do tamanho do instrumento de medida de que dispomos.

A ciência tradicional trabalha com modelos de curvas e superfícies lisas. À medida que as olhamos mais de perto, mais elas adquirem a forma plana — do mesmo modo que a superfície da terra é aproximadamente esférica e nós a percebemos como sendo chata.

Porém, como ficou claro no exemplo da costa rochosa, alguns objetos não se mostram planos nem quando examinados de muito perto. E existem muitos outros exemplos na natureza de configurações que possuem detalhes diferentes em cada dimensão de sua estrutura. Porém, apenas recentemente os cientis-





tas e matemáticos tomaram consciência da importância do estudo desses objetos e criaram modelos para suas estruturas — uma série de figuras geométricas denominadas *figuras de dimensões fracionadas*, ou *fractais*.

Um dos primeiros pesquisadores a estudá-las foi o norte-americano Benoit Mandelbrot que reconheceu nessas figuras um fecundo campo de trabalho, muito apropriado à concepção de modelos para objetos naturais irregulares.

MEDINDO UM OBJETO

Se você tomar dois pedaços de corda do mesmo tamanho e uni-los por uma das pontas, poderá dizer que obteve



uma cópia do pedaço original com o dobro do tamanho. No caso de uma folha de papel, porém, serão necessárias outras três folhas para se obter uma com o formato da original mas com o dobro da extensão; com uma barra de rapadura, você irá precisar de mais sete cubos.

Os números 2, 4 e 8 são parte de uma seqüência formada por meio da multiplicação do 2 por ele mesmo diversas vezes 2, 2x2 e 2x2x2, ou 2↑1, 2↑2 e 2↑3. A potência à qual está elevado o 2 equivale à dimensão do objeto.

Como vimos, para alcançar o dobro de seu tamanho, um objeto unidimensional (a corda) tem que ser multiplicado por dois; um objeto bidimensional (a folha) deve ser multiplicado por quatro e assim por diante. Suponhamos agora que, para duplicar o tamanho de certo objeto, sejam necessários três exemplares idênticos. Como esse número está entre dois (equivalente a uma dimensão) e quatro (correspondente a duas dimensões), deduzimos que a dimensão do objeto está entre 1 e 2.

À primeira vista, tal situação parece improvável — embora correta, em termos lógicos. Mas o estudo das dimensões fracionadas pode mesmo nos levar a resultados surpreendentes.

O matemático alemão Von Koch foi um dos primeiros estudiosos a desenhar um diagrama representando esse tipo de ocorrência: trata-se da chamada *curva floco de neve* (muito semelhante a um floco de neve visto ao microscópio). Cada um dos lados dessa curva é formado por quatro cópias dela mesma, com um terço do seu tamanho. A dimensão, nesse caso, é um pouco acima de 1,26. Várias formas da natureza, como a própria costa rochosa, podem ser vistas como uma figura dessas.

Como todos os modelos matemáticos, a curva de dimensão fracionada — de definição muito precisa — não é capaz de descrever com perfeição a grande diversidade de objetos naturais, limitando-se a uma aproximação. É, porém, muito útil na explicação de diferentes estruturas — de veias do corpo humano a montanhas, rios ou árvores.

AUTO-SEMELHANÇA

Voltemos um pouco ao exemplo da corda, do papel e da rapadura. Cada um desses objetos poderia ser formado por várias cópias menores idênticas. Estaríamos então diante de um caso de auto-semelhança, onde cada parte do objeto é uma cópia menor do original.

As figuras criadas por Von Koch constituem uma aplicação muito rígida desse princípio, o que explica a extrema regularidade de sua forma. Os exemplares naturais não apresentam necessariamente auto-semelhança — é fácil constatar que um pedaço da casca de uma árvore, por exemplo, não é uma pequena cópia da casca inteira. Entretanto, ele poderia estar em alguma outra parte da casca, pois todos os pedaços se parecem muito. Esse fenômeno, denominado *auto-semelhança estatística*, é muito comum na natureza.



DIMENSÕES FRACIONADAS E GRÁFICOS

Como tantos modelos matemáticos, esse tipo de curva pode ser programado em seu micro para simular a realidade, evidenciando algumas de suas características. No estudo dos fractais, especificamente, a computação gráfica tem sido amplamente utilizada, oferecendo a solução ideal para o problema da geração de figuras irregulares, tais como montanhas, mares e uma série de outros elementos da natureza.

A recursão constitui um excelente meio para a aplicação do princípio da auto-semelhança que envolve a construção de uma figura através de figuras idênticas menores. No caso, uma subrotina principal se encarregará de desenhar os vários elementos idênticos em diferentes tamanhos.

Se você quiser obter uma curva semelhante à curva floco de neve bastará escolher uma figura bem simples e repeti-la várias vezes, em diversas escalas, adicionando mais e mais detalhes a cada nível de tamanho. Experimente, por exemplo, tomar uma reta e aplicar a regra que prescreve: "toda linha reta deve ser substituída por um par de retas em ângulo reto".

Se você tentar fazer isso em uma folha de papel, verá que a cada passo mais detalhes são introduzidos no desenho.

O programa que apresentamos a seguir faz uma demonstração desse processo. Como todos os gráficos elaborados conforme o modelo de dimensão fracionada, nosso desenho poderia se estender infinitamente. Porém, como não temos uma capacidade gráfica muito grande, os detalhes nos passariam despercebidos, da mesma maneira que as irregularidades de um grão de areia. Portanto, o programa irá parar assim que alcançar um certo nível de recursão.



```
10 BORDER 0: BRIGHT 1: PAPER
0· INK 7· CLS
20 LET mn=2
30 LET c=PI/180
40 LET l=120  LET x=70· LET y
=50  LET an PI/2
45 PLOT  INVERSE 1: OVER 1;x,
y
50 GOSUB 1000
80 STOP
1000 LET l=l/1.414
1010 IF l<mn THEN  LET . l*1 41
4· LET x=x+(l*SIN (an)): LET y=
y (l*COS (an))  DRAW x-PEEK 236
77,y-PEEK 23678  RETURN
1020 LET an an+PI/4  GOSUB 1000
1030 LET an an PI/2  GOSUB 1000
1040 LET an=an+PI/4· LET l l*1
414  RETURN
```



```
10 PMODE 4,1 PCLS:SCREEN 1,1
20 MN=2
30 C=ATN(1)/45·PI=4*ATN(1)
40 L=120:X=70 Y=140 AN=PI/2
45 LINE (X,Y),PRESET
50 GOSUB 1000
60 GOTO 60
1000 L=L/1 414
1010 IF L<MN THEN L=L*1.414:X=X
+(L*SIN(AN)) Y=Y+(L*COS(AN)):LI
NE-(X,Y),PSET·RETURN
1020 AN=AN+PI/4 GOSUB 1000
1030 AN=AN-PI/2 GOSUB 1000
1040 AN=AN+PI/4:L=L*414:RETURN
```



```
10  HGR2
20 MN = 2
30 C =  ATN (1) / 45:PI = 4 *
ATN (1)
40 L = 120·X = 70:Y = 140:AN =
PI / 2
45  HPLOT X,Y TO X,Y
50  GOSUB 1000
60  GOTO 60
1000 L = L / 1 414
1010  IF L < MN THEN L = L * 1.
414:X = X + (L *  SIN (AN)):Y =
 Y + (L *  COS (AN)): HCOLOR= 3
· HPLOT  TO X,Y· RETURN
1020 AN = AN + PI / 4· GOSUB 10
00
1030 AN = AN - PI / 2· GOSUB 10
00
1040 AN = AN + PI / 4·L = L * 1
 414· RETURN
```



```
10 SCREEN 2
20 MN 2
30 C=ATN(1)/45·PI=4*ATN(1)
40 L=120:X=70:Y=140:AN=PI/2
45 LINE (X,Y),4
50 GOSUB 1000
60 GOTO 60
1000 L=L/1 414
1010 IF L<MN THEN L=L*1 414·X=X
+(L*SIN(AN)):Y=Y+(L*COS(AN)):LI
NE  (X,Y),11·RETURN
1020 AN=AN+PI/4·GOSUB 1000
1030 AN AN PI/2·GOSUB 1000
1040 AN=AN+PI/4·L-L*1 414·RETUR
N
```

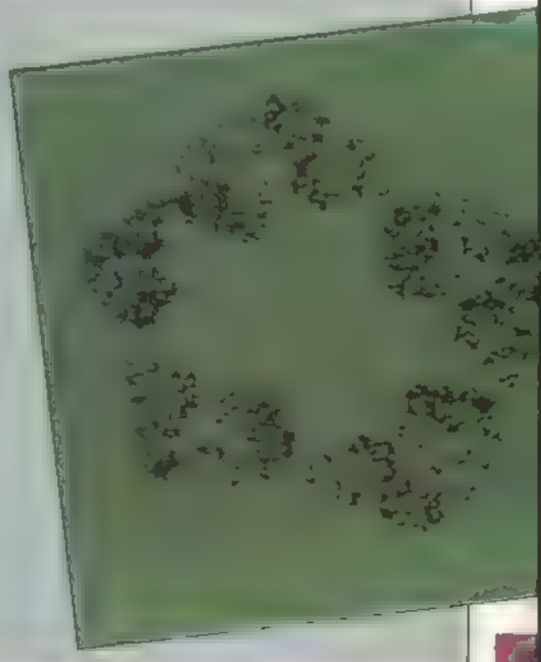
O programa desenhará na tela uma curva em forma de C. Inicialmente, ele traça uma reta, que logo é substituída por um par de retas em ângulo reto. A sub-rotina entre as linhas 1000 e 1040, que usa **SIN** e **COS** para construir os ângulos necessários, é chamada na linha 50. Em um processo recursivo, ela chama a si mesma repetidamente, para substituir as linhas retas por pares de retas perpendiculares.

A variável da linha 20 contém o comprimento da linha mais curta — a recursão será encerrada quando alcançar esse valor. Tente mudá-lo e observe o efeito resultante: valores menores aumen-

MODELOS EXPERIMENTAIS

A principal diferença entre as figuras de dimensão fracionada geradas no computador e as formas da natureza está na irregularidade e desordem destas últimas.

No próximo artigo sobre fractais, veremos como acrescentar elementos aleatórios ao processo de construção das figuras, aproximando-as mais dos modelos naturais. Um dos programas, um gerador de figuras, permite que se introduza a semente geradora do desenho — ou seja, uma infinidade de formas poderá ser criada simplesmente variando-se a informação inicial. Assim, a cada nova execução do programa, você obterá uma curva diferente. Uma das muitas possibilidades é mostrada na ilustração a direita



tam o número de níveis de recursão e, em consequência, o tempo de execução. Mas, se introduzirmos valores maiores, o programa será acelerado e poderemos visualizar melhor o processo de elaboração do gráfico.

Se você acompanhar o desenvolvimento da figura na tela com atenção, verificará que cada parte da curva se parece com a própria curva (princípio da auto-semelhança). Apesar disso, não é fácil antever, logo no início, a forma que ela irá adquirir.

CURVA DO DRAGÃO

Quando trabalhamos com curvas de dimensão fracionada, poucas linhas de programa são necessárias para produzir uma bela figura — sobretudo se utilizarmos a recursão. Digite e execute o próximo programa para obter uma curva em forma de dragão.

No programa anterior, cada reta era substituída por um par de retas perpendiculares, traçadas sempre em uma mesma direção. Neste programa, porém, pares de retas são colocados alternadamente de um e de outro lado de cada reta, até formar a figura final.

S

```
10 BORDER 0: PAPER 0· INK 7:
CLS : CLEAR 30000: LET S=
32767· DEF FN A(X)=(X/8-INT (
X/8))*8
20 LET MN=1· LET A=0
30 LET C=ATN (1)/45: DIM S(10
)· DIM C(10)
40 FOR I=1 TO 8: LET S(I)=SIN
A
50 LET C(I)=COS A: LET A=A+PI
/4· NEXT I
60 LET L=128: LET X=52: LET Y
=80· LET T=-1· POKE S,T+1·
LET S=S 1
65 DRAW  INVERSE 1; OVER 1;X-
PEEK 23677,Y-PEEK 23678
70 GOSUB 1000
80 STOP
1000 LET L=L/1.414
1010 IF L<MN THEN  LET L=L*1.41
4: LET X=X+(L*C(I)): LET Y=Y-(L
*S(I)): DRAW X-PEEK 23677,Y-PEE
K 23678· RETURN
1020 LET I=FN A(I+T): POKE S,T+
1: LET S=S-1: LET T=1· GOSUB 10
00: LET S=S+1: LET T=(PEEK S)-1
1030 LET I=FN A(I-2*T): POKE S,
T+1: LET S=S-1: LET T=-1: GOSUB
 1000
1040 LET S=S+1: LET T=(PEEK S)-
1: LET I=FN A(I+T): LET L=L*1.4
14· RETURN
```

T

```
10 PMODE 4:PCLS:SCREEN 1,1:CLEA
R 200,30000:S=32767
20 MN=1
30 C=ATN(1)/45:PI=4*ATN(1)
40 FOR I=0 TO 7:S(I)=SIN(A)
50 C(I)=COS(A):A=A+PI/4:NEXT
60 L=128:X=52:Y=80:T=-1:POKE S,
T+1:S=S-1
65 LINE -(X,Y),PRESET
70 GOSUB 1000
80 GOTO 80
1000 L=L/1.414
1010 IF L<MN THEN L=L*1.414·X=X
+(L*C(I))·Y=Y-(L*(S(I)):LINE-(X
,Y),PSET:RETURN
1020 I=(I+T) AND 7:POKE S,T+1:S
=S-1:T=1:GOSUB 1000:S=S+1:T=PEE
K(S)-1
1030 I=(I-2*T)AND7:POKE S,T+1:S
=S-1·T=-1:GOSUB 1000
1040 S=S+1:T=PEEK(S)-1:I=(I+T)
AND 7:L=L*1.414:RETURN
```

```
10 S = 900
20  HGR2 :MN = 1
30 C =  ATN (1) / 45·PI = 4 *
ATN (1)
40  FOR I = 0 TO 7:S(I) =  SIN
(A)
50 C(I) =  COS (A)·A = A + PI /
 4: NEXT
60 L = 128·X = 52:Y = 80:T =  -
 1· POKE S,T + ] S = S   .
65  HCOLOR= 3: HPLOT X,Y TO X,Y

70  GOSUB 1000
80  GOTO 80
1000 L = L / 1.414
1010  IF L < MN THEN L = L * 1
414:X = X + (L * C(I)):Y = Y -
(L * S(I))· HPLOT  TO X,Y  RETU
RN
1020 I = ((I + T) - 8 *  INT ((
I + T) / 8)): POKE S,T + 1·S =
S - 1:T = 1: GOSUB 1000:S = S +
 1·T =  PEEK (S)   1
1030 I = ((I - 2 * T)   8 *  IN
T ((I - 2 * T) / 8)): POKE S,.
+ 1:S = S - 1:T =  - 1: GOSUB .
000
1040 S = S + 1:T =  PEEK (S)
1:I = ((I + T) AND 7):L = L * 1
.414· RETURN
```

```
10 SCREEN 2·CLEAR 200,50000'·S=
52767'
20 MN=1
30 C=ATN(1)/45·PI=4*ATN(1)
40 FOR I=0 TO 7·S(I)=SIN(A)
50 C(I)=COS(A):A=A+PI/4·NEXT
60 L=128:X=52:Y=80:T=-1:POKE S,
T+1:S=S-1
65 LINE -(X,Y),4
70 GOSUB 1000
80 GOTO 80
1000 L=L/1.414
1010 IF L<MN THEN L=L*1.414·X=X
+(L*C(I)):Y=Y-(L*S(I))·LINE -(X
,Y),11·RETURN
1020 I=(I+T) AND 7:POKE S,T+1:S
=S-1 T=1·GOSUB 1000·S=S+1·T PEE
K(S)-1
1030 I (I 2*T) AND 7·POKE S,T+1
·S=S-1·T=-1·GOSUB 1000
1040 S=S+1·T=PEEK(S)-1:I=(I+T)A
ND 7:L=L*1.414:RETURN
```



Existe alguma aplicação prática para os fractais?

Os fractais são uma descoberta relativamente recente da matemática e das ciências da computação. Assim, há muito a ser investigado acerca de seu funcionamento e as aplicações práticas ainda são bastante restritas. Os fractais oferecem aos artistas e cientistas a possibilidade de representar as formas irregulares, típicas da natureza e já têm marcado sua presença no mundo das artes gráficas, cinema, televisão etc. Grandes empresas de televisão, por exemplo, estão utilizando fractais randômicos para gerar em computador fantásticas paisagens e cenários, terrestres ou extraterrestres. A simulação fractal apresenta a vantagem de, com pequenas mudanças nos parâmetros do programa, produzir efeitos totalmente diferentes na tela.

Outra aplicação prática interessante dos fractais é o desenho de cenários complexos em simuladores de vôo. A Força Aérea norte-americana já dispõe de programas que permitem gerar o cenário de uma batalha aérea entre montanhas escarpadas, sobre desertos etc., para treinar o piloto de helicópteros ou caças.

Esse programa difere do anterior quanto à regra de formação do desenho e, também, quanto à velocidade de execução. Em vez de elaborar o ângulo cada vez que um elemento é traçado, o programa calcula o seno e o co-seno dos ângulos (linhas 40 e 50) e coloca os valores obtidos em duas matrizes. Recuperar esses valores em uma matriz é bem mais rápido que calculá-los sempre que uma linha é reconstruída.

A curva em C e a curva do dragão não são apenas curiosidades matemáticas. Além de muito bonitas, elas constituem um interessante objeto de estudo. Procure, por exemplo, alterar alguns valores dentro dos dois programas. Você obterá, então, um estoque inesgotável de novas formas.

DOMINE O VÍDEO DO MSX

- TELA DE QUARENTA COLUNAS
- VPEEK E VPOKE
- FORMATO DAS LETRAS
- TABELA DE PADRÕES
- NOVOS CARACTERES

O MSX difere dos demais micros por possuir um microprocessador e uma memória de dezesseis kbytes dedicados só ao controle da tela. Desvende os segredos da VRAM e sinta a diferença!

Muitos usuários dos micros MSX enfrentam dificuldades em utilizar comandos como **BASE**, **VPEEK** e **VPOKE**. Para melhor aproveitar esses recursos da linguagem BASIC, temos inicialmente que entender como é organizada a memória de vídeo — VRAM —, um item desprezado pela maioria dos manuais.

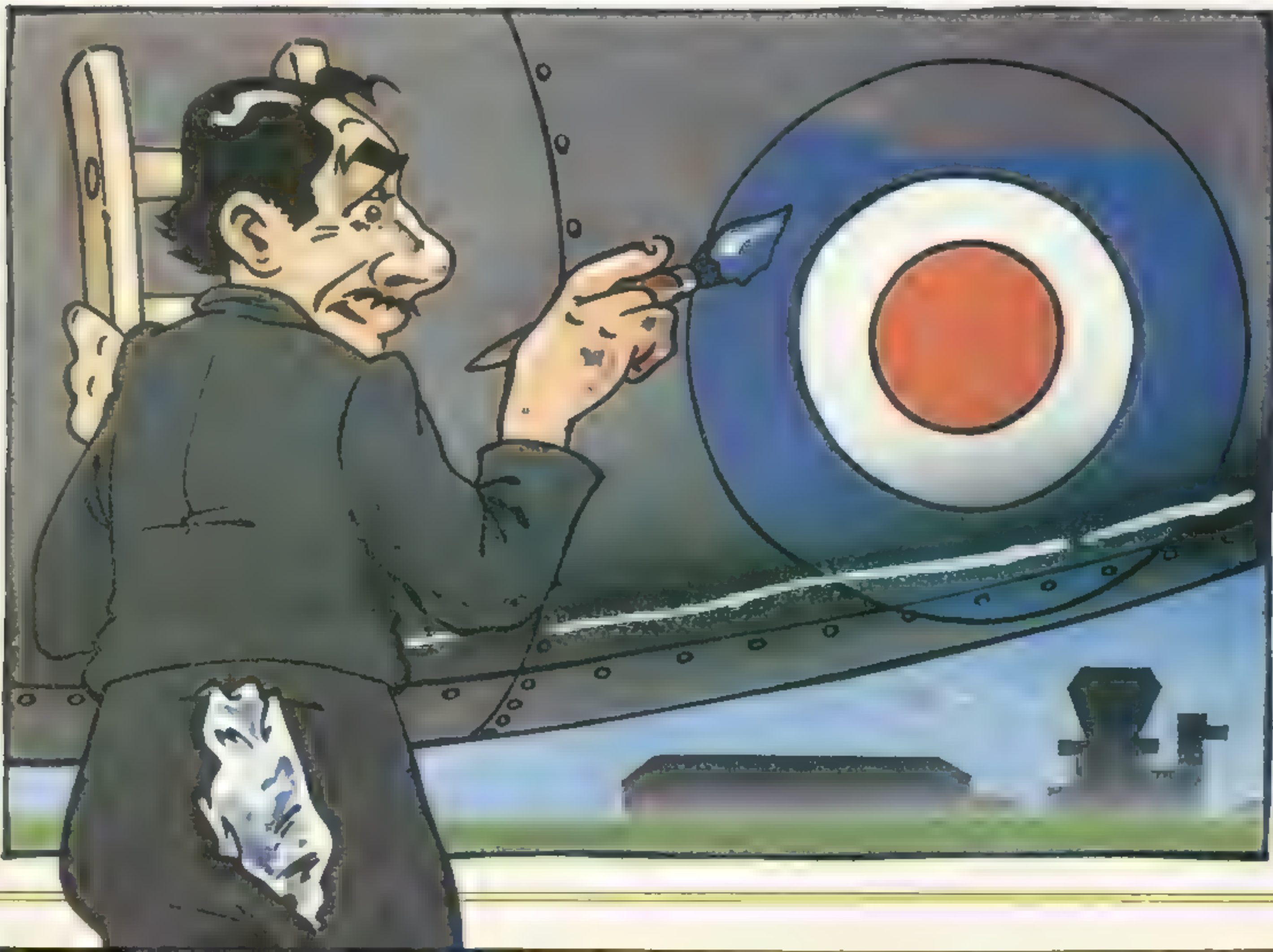
O MSX possui um chip independente da unidade central de processamento (CPU) destinado exclusivamente ao controle da tela. Conhecido como VDP (*Video Display Processor*), ele gera imagens a partir de dados armazenados nos dezesseis kbytes de uma memória independente da RAM (memória de acesso direto, onde ficam armazenados o programa BASIC e as variáveis). Dessa forma, a utilização de gráficos multicoloridos de alta resolução não compromete a quantidade de memória disponível, como acontece com outros micros.

TELA DE QUARENTA COLUNAS

Ao ligarmos um microcomputador da linha MSX, observamos uma tela de textos em que cada linha acomoda, teoricamente, quarenta caracteres. Se isso não ocorrer em seu micro, digite o comando **SCREEN 0**.

A tela — quer contenha textos, listagens de programas ou símbolos gráficos — é simplesmente uma reprodução da parte da VRAM denominada *Tabela de Nomes* (TN). Essa área da memória de vídeo possui 960 bytes de comprimento, correspondentes aos 960 caracteres que podem ser mostrados na tela simultaneamente (são 24 linhas com quarenta caracteres cada uma).

Cada uma dessas posições pode conter um número qualquer entre 0 e 255, número que é interpretado pelo VDP como o código ASCII (sigla de *American Standard Code for Information Interchange*) de um caractere.



VPEEK E VPOKE

Em qualquer micro, o comando **POKE** do BASIC nos permite armazenar um número entre 0 e 255 numa posição da RAM. Da mesma forma, o conteúdo de qualquer endereço da RAM ou da ROM é obtido com o comando **PEEK**.

Como a VRAM não depende da RAM, necessitamos de comandos que nos possibilitem ler ou escrever bytes diretamente na memória de vídeo, ou seja, **VPEEK** e **VPOKE**. Experimente essa nova forma de escrever na tela:

```
10 CLS
20 VPOKE 417,77
30 VPOKE 419,83
40 VPOKE 421,88
50 END
```

Esse programa coloca diretamente nas posições 417, 419 e 421 os códigos das letras M, S e X. O VDP transforma o código da Tabela de Nomes no caractere correspondente, que é imediatamente reproduzido na tela.

Para conferir, digite:

```
PRINT VPEEK (417)
PRINT VPEEK (419)
PRINT VPEEK (421)
```

Estes comandos diretos recuperam os códigos das letras M, S e X diretamente da Tabela de Nomes. Se você não gosta muito de códigos, obtenha as letras correspondentes utilizando:

```
PRINT CHR$(VPEEK (417))
PRINT CHR$(VPEEK (419))
PRINT CHR$(VPEEK (421))
```

Depois de familiarizado com o funcionamento desses dois comandos, o usuário poderá ter acesso às quarenta colunas da tela (o que não é possível por intermédio do comando **PRINT**). Muitos leitores já devem ter observado que o texto começa a ser escrito a partir da segunda coluna, reduzindo, portanto, o espaço total em uma coluna.

Para ilustrar isso, note que

```
LOCATE 0,10:PRINT "*"
```

imprime um asterisco na segunda coluna, deixando um espaço vazio à esquerda, enquanto os comandos

```
LOCATE 38,10:PRINT "*"
```

e

```
LOCATE 39,11:PRINT "*"
```

imprimem asteriscos na mesma coluna.

Use o seguinte programa para ocupar a primeira coluna da tela:

```
10 SCREEN 0:KEY OFF
20 FOR I=0 TO 959 STEP 40
30 VPOKE BASE(0)+I,207
40 NEXT
```

A linha 10 seleciona a tela de textos e apaga os rótulos das teclas de função da parte inferior do vídeo. Um laço **FOR..NEXT** entre as linhas 20 e 40 coloca várias vezes o caractere de código 207 na primeira posição de cada linha. Deve-se ressaltar que, apesar de termos usado a instrução **BASE(0)** na linha 30, ela é perfeitamente dispensável, uma vez que seu valor é zero.

COMO SE DESENHAM AS LETRAS

Vimos até agora que, ao se colocar um número dentro da Tabela de Nomes, o VDP imediatamente o interpreta como o código ASCII de uma letra ou de um símbolo. Como o processador sabe o formato de cada uma das letras do alfabeto?

O leitor já deve ter notado que as letras que aparecem no vídeo da TV são compostas por pequenos pontos. Cada caractere tem seu perfil — ou padrão — desenhado em um quadrado com oito pontos de lado. Este padrão é codificado em números binários (formados por zeros e uns), a fim de que o computador possa compreendê-lo. Se fizermos com que os "uns" correspondam aos pontos acesos e os "zeros", aos apagados, poderemos codificar todos os perfis. Como um número binário entre 0 e 255 tem no máximo oito algarismos, cada caractere será representado através de uma série de oito números binários menores que 255, ou seja, oito bytes.

Para que o VDP reconheça o formato dos caracteres, uma porção da VRAM é separada para armazenar os 2048 bytes necessários para representar os 256 símbolos existentes (256*8 = 2048). Esta parte da memória de vídeo é chamada de Tabela de Padrões e começa no endereço 2048 da VRAM.

BASE

Para que o usuário não confunda os endereços da RAM e da VRAM, o BASIC do MSX oferece a instrução **BASE**, que evita a memorização dos endereços e permite modificar a posição das tabelas. Assim, para a tela de textos de quarenta colunas, o endereço inicial da Tabela de Nomes está em **BASE (0)** e o da Tabela de Padrões em **BASE (2)**. Confira:

```
PRINT BASE(0)
```

e

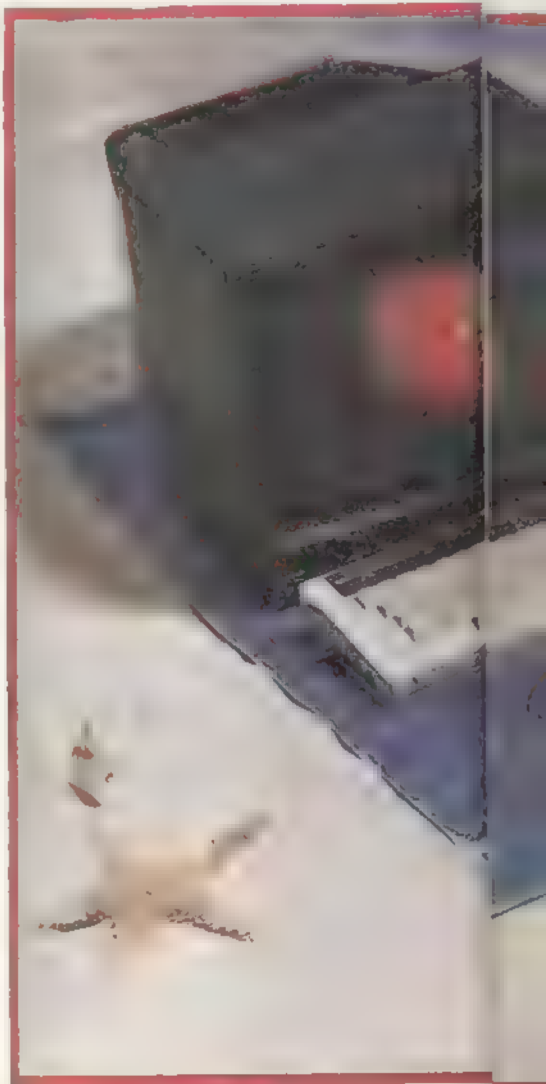
```
PRINT BASE(2)
```

TABELA DE PADRÕES

Visando a uma melhor compreensão da Tabela de Padrões, mostraremos um pequeno programa que nos facilitará o acesso a essa região da VRAM.

Ao ser executado, o programa imprime na tela a porção da tabela que corresponde a um caractere. São impressos também os endereços (com conteúdos em números decimal e binário) e o caractere com seu código. Dessa forma, o leitor entenderá como os números binários codificam o formato dos caracteres.

```
10 Z$="00000000"
20 SCREEN 0:KEY OFF
30 J=520
40 GOSUB100
50 K$=INKEY$:IF K$="" THEN 50
60 IF K$=CHR$(31) AND J<2040 THEN J=J+8
70 IF K$=CHR$(30) AND J>7 THEN J=J-8
```



```
80 GOTO 40
100 LOCATE 12,1:PRINT "CARACTER: "
110 VPOKE 64,J/8
120 LOCATE 10,3:PRINT "Código ASCII  ",J/8
130 LOCATE 1,6:PRINT "Endereço",TAB(14),"Conteudo";TAB(27),"Conteúdo"
140 PRINT:PRINT TAB(3);"VRAM";TAB(14);"Decimal";TAB(27);"Binário"
150 PRINT
160 FOR I=J TO J+7
170 PRINT TAB(2);BASE(2)+I;TAB(15);VPEEK(BASE(2)+I);TAB(27);RIGHT$(Z$+BIN$(VPEEK(BASE(2)+I)),8)
180 NEXT
190 RETURN
```

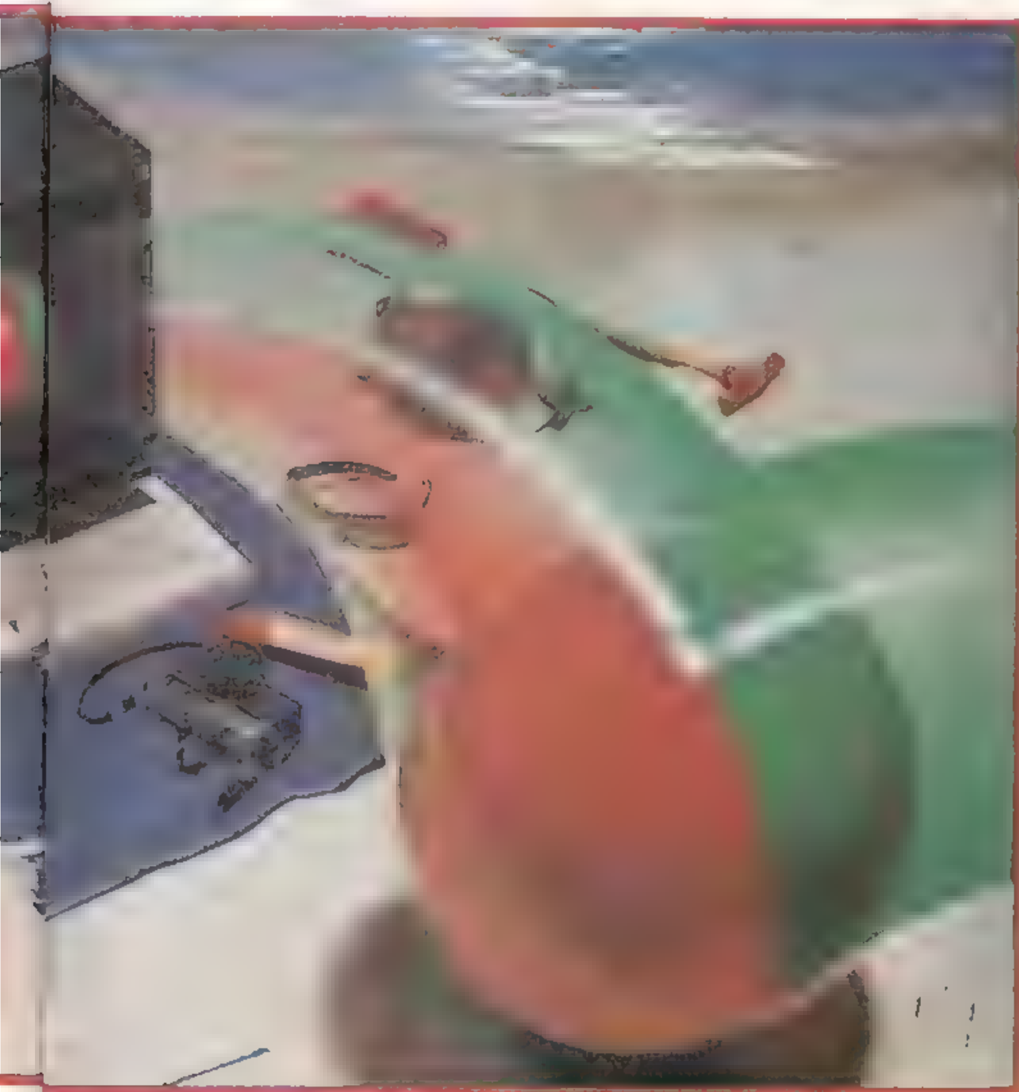
A linha 10 cria uma variável auxiliar que permite a conversão decimal/binário. A linha 20 seleciona a tela de quarenta colunas e apaga o rodapé com as teclas de função.

A variável **J** controla a porção da Tabela de Padrões mostrada na tela, definindo o caractere que tem seu padrão exibido. Seu valor inicial, determinado na linha 30, faz com que a primeira letra impressa seja o 'A' — cujo código é 65 (65*8 = 520). A linha 40 chama a sub-rotina que imprime a porção da tabela que nos interessa. As linhas 50 e 80 permitem avançar ou retroceder dentro da Tabela de Padrões usando as teclas de controle do cursor.

A sub-rotina que cuida da impressão dos dados na tela vai da linha 100 à 190. As linhas 100 e 110 imprimem o caractere em questão; a linha 120 imprime seu código e as linhas 130 a 150 encarregam-se do cabeçalho.

O laço **FOR..NEXT**, da linha 160 à 180, imprime o conteúdo dos oito bytes que determinam o padrão do caractere e seus endereços na VRAM.

Depois de uma ligeira exploração da Tabela de Padrões, o leitor notará que os caracteres gráficos (aqueles que não são letras ou sinais de pontuação) surgem truncados no vídeo. A razão disso é a incapacidade do MSX de imprimir mais que 256 pontos horizontais. Essa limitação fica mais evidente quando precisamos de uma resolução de 320 pontos na horizontal para representar quarenta caracteres, cada um com oito pontos de largura. Assim, quando estamos em **SCREEN 0**, somente são mostrados os seis pontos — ou bits — mais à esquerda do padrão, suficientes para definir as letras mas não os caracteres gráficos, que muitas vezes utilizam até oito pontos.



UM NOVO CONJUNTO DE CARACTERES

O formato das letras é definido na VRAM, e nada nos impede de modificá-lo, já que dispomos do comando **VPOKE** para executar tal tarefa. Confira:

```
10 FOR I=65*8 TO 91*8-1
20 READ A:VPOKE BASE(2)+I,A
30 NEXT
40 FOR I=97*8 TO 123*8-1
50 READ A:VPOKE BASE(2)+I,A
60 NEXT
70 FOR I=48*8 TO 58*8-1
80 READ A:VPOKE BASE(2)+I,A
90 NEXT
100 END
1000 DATA 120,72,72,200,248,200,200,0
1010 DATA 112,80,80,248,200,200,240,0
1020 DATA 120,72,64,192,192,200,248,0
1030 DATA 112,72,72,200,200,200,240,0
1040 DATA 120,64,64,240,192,192,248,0
1050 DATA 120,64,64,240,192,192,192,0
1060 DATA 120,64,64,216,200,200,248,0
1070 DATA 72,72,72,248,200,200,200,0
1080 DATA 16,16,16,48,48,48,48,0
1090 DATA 48,16,16,24,152,152,120,0
1100 DATA 72,72,80,224,208,200,200,0
1110 DATA 64,64,64,192,192,192,248,0
1120 DATA 68,108,84,196,196,196,196,0
1130 DATA 72,104,104,216,216,216,200,0
1140 DATA 120,72,72,200,200,200,248,0
1150 DATA 120,72,72,248,192,192,192,0
1160 DATA 120,72,72,200,200,216,248,4
1170 DATA 112,72,72,248,208,200
```

```
,200,0
1180 DATA 112,80,64,120,24,152,
248,0
1190 DATA 248,32,32,32,96,96,96
,0
1200 DATA 72,72,72,200,200,200,
248,0
1210 DATA 200,200,200,200,200,8
0,32,0
1220 DATA 196,196,212,212,212,4
0,40,0
1230 DATA 80,80,80,32,208,200,2
00,0
1240 DATA 136,136,136,80,48,48,
48,0
1250 DATA 120,72,16,224,192,200
,248,0
1260 DATA 0,0,112,16,240,208,24
8,0
1270 DATA 128,128,128,240,200,2
00,248,0
1280 DATA 0,0,112,64,192,192,24
8,0
1290 DATA 16,16,112,80,208,208,
248,0
1300 DATA 0,0,112,80,240,192,24
8,0
1310 DATA 48,80,64,64,240,192,1
92,0
1320 DATA 0,240,144,224,64,248,
200,248
1330 DATA 64,64,64,248,200,200,
200,0
1340 DATA 16,0,16,16,48,48,48,0
1350 DATA 16,0,16,16,24,152,152
,120
1360 DATA 0,64,72,80,224,208,20
0,0
1370 DATA 32,32,32,96,96,96,112
,0
1380 DATA 0,0,128,208,168,168,1
68,0
1390 DATA 0,0,64,112,200,200,20
0,0
1400 DATA 0,0,112,200,200,200,1
12,0
1410 DATA 0,0,120,72,200,248,19
2,192
1420 DATA 0,0,240,144,144,248,2
4,24
1430 DATA 0,0,80,104,192,192,19
2,0
1440 DATA 0,0,112,64,56,136,248
,0
1450 DATA 32,32,112,32,32,96,11
2,0
1460 DATA 0,0,72,72,200,200,248
,0
1470 DATA 0,0,200,200,200,80,32
,0
1480 DATA 0,0,136,168,168,168,8
0,0
1490 DATA 0,0,80,80,32,208,200,
0
1500 DATA 0,0,72,72,48,48,48,0
1510 DATA 0,0,120,8,112,96,120,
0
1520 DATA 120,72,72,216,232,200
,248,0
1530 DATA 32,96,32,48,48,48,48,
0
1540 DATA 120,72,8,248,192,200,
248,0
1550 DATA 112,16,16,120,24,152,
248,0
1560 DATA 144,144,144,248,24,24
,24,0
1570 DATA 240,128,128,248,24,24
,248,0
1580 DATA 112,64,64,248,200,200
,248,0
1590 DATA 224,32,32,48,48,48,48
,0
1600 DATA 112,80,80,248,152,152
,248,0
1610 DATA 240,144,144,248,24,24
,248,0
```

O conjunto de letras futuristas permanece à disposição do usuário até que o micro seja desligado ou receba um comando **SCREEN**.

O programa é composto de três laços **FOR...NEXT** e um grande número de linhas **DATA**, cada uma com oito bytes que definem o padrão dos símbolos.

O primeiro laço — que vai da linha 10 a 30 — modifica os padrões das letras maiúsculas (códigos ASCII de 65 a 90). O segundo laço — linhas 40 a 60 — cuida das minúsculas (códigos 97 a 122) e o terceiro — linhas 70 a 90 — dos números (48 a 57). Não incluímos caracteres acentuados para não estender muito as linhas **DATA**. Entretanto, se o leitor compreender bem o funcionamento do programa, não terá dificuldade em criá-los, completando assim o conjunto.

O programa que explora a Tabela de Padrões poderá ser aplicado para que se conheça mais detalhadamente os caracteres recém-criados.

Em contraste com o traço de vanguarda desses símbolos artificiais, criamos um segundo conjunto, dessa vez em letras de fôrma. As próximas linhas deverão ser acrescentadas ao programa anterior, formando um maior, que será completado no final do artigo.

```
5 RESTORE 1620
1620 DATA 16,40,72,72,120,72,13
2,0
1630 DATA 120,36,36,56,36,68,24
8,0
1640 DATA 24,36,64,64,64,68,56,
0
1650 DATA 120,36,36,36,36,68,24
8,0
1660 DATA 24,36,32,56,64,68,56,
0
1670 DATA 60,72,16,16,60,144,96
,0
1680 DATA 24,36,64,68,68,60,136
,112
1690 DATA 68,72,72,72,120,72,13
2,0
1700 DATA 8,24,40,8,8,16,224,0
1710 DATA 60,72,8,8,8,136,144,9
6
1720 DATA 68,72,80,96,80,72,132
,0
1730 DATA 32,64,64,64,64,68,248
,0
1740 DATA 40,84,84,84,84,84,132
,0
1750 DATA 68,68,100,84,76,68,13
2,0
1760 DATA 24,36,68,68,68,72,48,
0
1770 DATA 120,36,36,36,120,32,1
92,0
1780 DATA 24,36,68,68,116,72,52
,0
1790 DATA 120,36,36,36,120,40,1
96,0
1800 DATA 24,36,64,56,4,136,112
,0
1810 DATA 124,16,16,16,16,32,19
2,0
1820 DATA 72,72,72,72,72,72,52,
0
1838 DATA 76,72,72,72,72,80,32,
0
1840 DATA 68,84,84,84,84,84,40,
0
1850 DATA 68,36,40,16,40,72,132
,0
1860 DATA 68,36,36,36,24,16,224
,0
1870 DATA 60,68,8,124,32,68,248
,0
1880 DATA 0,0,56,72,136,136,116
,0
1890 DATA 32,64,64,120,68,68,24
8,0
1900 DATA 0,0,24,36,64,64,188,0
1910 DATA 4,4,28,36,68,68,188,0
1920 DATA 0,0,56,68,72,48,220,0
1930 DATA 8,20,32,32,32,252,32,
32
1940 DATA 0,0,60,68,68,184,8,11
2
1950 DATA 32,32,40,52,36,100,16
4,0
1960 DATA 16,0,16,16,16,48,204,
0
1970 DATA 8,0,24,40,72,140,8,11
2
1980 DATA 16,32,36,40,48,40,196
,0
1990 DATA 16,40,40,40,40,16,236
,0
2000 DATA 0,0,40,84,84,84,132,0
2010 DATA 0,0,88,100,68,68,132,
0
2020 DATA 0,0,28,36,92,68,184,0
2030 DATA 0,0,56,36,36,252,32,3
2
2040 DATA 0,0,56,68,68,188,4,4
2050 DATA 0,0,32,60,36,68,132,0
2060 DATA 0,32,48,40,36,68,152,
0
2070 DATA 36,16,124,16,16,48,20
4,0
2080 DATA 0,0,68,68,68,68,188,0
2090 DATA 0,0,76,72,72,80,160,0
2100 DATA 0,0,196,84,84,84,40,0
2110 DATA 0,0,68,40,16,40,196,0
2120 DATA 0,0,36,36,36,252,8,11
2
2130 DATA 0,0,124,4,56,64,252,0
2140 DATA 24,36,76,84,100,72,48
,0
2150 DATA 8,24,8,16,16,32,112,0
2160 DATA 56,68,36,8,48,100,88,
```



```
0
2170 DATA 24,36,4,24,68,68,56,0
2188 DATA 4,12,24,40,72,124,16,
0
2190 DATA 60,32,64,120,4,4,120,
0
2200 DATA 12,16,32,120,68,68,56
,0
2210 DATA 60,4,8,16,32,64,64,0
2220 DATA 56,68,68,56,68,68,56,
0
2230 DATA 24,36,68,68,56,8,112,
0
```

O programa utilizado é o mesmo, com alterações apenas nas linhas **DATA**. O comando **RESTORE 1620** faz com que o comando **READ** inicie a leitura a partir da linha **DATA** 1620.

MODIFICANDO O VALOR-BASE

A VRAM tem 16384 bytes de comprimento, dos quais utilizamos apenas 3008 quando estamos no modo **SCREEN 0** (2048 para a Tabela de Padrões e 960 para a de nomes). Esse espaço ocioso pode ser aproveitado para armazenar conjuntos de caracteres e cópias da tela. O segredo do processo consiste na atribuição de novos valores para as instruções **BASE(0)** e **BASE(2)**.

As linhas seguintes devem ser adicionadas aos dois programas anteriores, formando, assim, um terceiro. É necessário apagar as linhas 5 a 100 com **DELETE 5-100.** Além disso, o programa precisa ser gravado antes de ser testado, a fim de permitir correções posteriores, no caso de ter ocorrido qualquer erro de digitação.

```
10 SCREEN 3:SCREEN 0:KEY OFF
20 FOR I=0 TO 2047
30 A=VPEEK(BASE(2)+I)
40 VPOKE 4096+I,A
50 VPOKE 6144+I,A
60 NEXT
100 FOR I=65*8 TO 91*8-1
110 READ A:VPOKE 4096+I,A
120 NEXT
130 FOR I=97*8 TO 123*8-1
140 READ A:VPOKE 4096+I,A
150 NEXT
160 FOR I=48*8 TO 58*8-1
170 READ A:VPOKE 4096+I,A
180 NEXT
200 FOR I=65*8 TO 91*8-1
210 READ A:VPOKE 6144+I,A
220 NEXT
230 FOR I=97*8 TO 123*8-1
240 READ A:VPOKE 6144+I,A
250 NEXT
260 FOR I=48*8 TO 57*8-1
270 READ A:VPOKE 6144+I,A
280 NEXT
300 J=0:GOSUB 360
310 FOR J=8192 TO 16300 STEP 10
24
320 BASE(0)=J
330 GOSUB 360
340 NEXT J
350 GOTO 420
360 FOR I=0 TO 959
365 IF I>439 AND I<560 THEN VPO
KE BASE(0)+I,32:NEXT
370 VPOKE BASE(0)+I,194+J/1024
380 NEXT: A$="Esta Tela Se Inic
ia Em"+STR$(J)
390 FOR I=1 TO LEN(A$)
400 VPOKE BASE(0)+486+I,ASC(MID
$(A$,I,1))
410 NEXT:RETURN
420 BASE(0)=0:BASE(2)=2048
430 K$=INKEY$:IF K$="" THEN 430
440 IF K$=CHR$(31) AND BASE(0)>
8192 THEN BASE(0)=BASE(0)-1024:
GOTO 430
450 IF K$=CHR$(31) AND BASE(0)=
8192 THEN BASE(0)=0:GOTO 430
460 IF K$=CHR$(30) AND BASE(0)>
0 AND BASE(0)<15000 THEN BASE(0
)=BASE(0)+1024:GOTO 430
```

```
470 IF K$=CHR$(30) AND BASE(0)=
0 THEN BASE(0)=8192:GOTO 430
480 IF K$=CHR$(29) AND BASE(2)>
2048 THEN BASE(2)=BASE(2)-2048:
GOTO 430
490 IF K$=CHR$(28) AND BASE(2)<
6144 THEN BASE(2)=BASE(2)+2048:
GOTO 430
500 IF K$=CHR$(27) THEN BASE(0)
=0:BASE(2)=2048:END
510 GOTO 430
```

O objetivo desse programa é mostrar como usar toda a área da VRAM mesmo em **SCREEN 0**. Ao ser executado, o programa gasta um longo período lendo as linhas **DATA** do final da listagem. A seguir observamos que o vídeo é preenchido sucessivas vezes com caracteres diversos. Uma mensagem aparece no centro de cada tela, mas é apagada antes que possa ser lida. Isso acontece porque o programa está criando, na parte ociosa da VRAM, "telas secundárias" que, de certa forma, coexistem com as demais telas. Elas podem ser recuperadas logo após a parada do programa através das teclas do cursor.

O programa também já tem prontas três cópias da Tabela de Padrões, cada qual com um tipo de caractere, o que nos permite mudar o aspecto das letras recorrendo às teclas do cursor.

Para interromper o programa, use a tecla **<ESC>**. Se o programa for interrompido dessa maneira, ou porque ocorreu alguma mensagem de erro, e ele estiver em uma tela diferente da original, o cursor de texto não retornará à tela e o programa parecerá estar bloqueado. Para fazer o micro voltar ao normal, digite às cegas (nada aparecerá no vídeo) o comando abaixo: **SCREEN 0**

```
SCREEN 0
```

e pressione a tecla **<ENTER>**.

COMO FUNCIONA

Na linha 10, selecionamos o modo de textos e apagamos o rodapé da tela. O laço **FOR...NEXT** entre as linhas 20 e 60 cria duas cópias da Tabela de Padrões, que começa normalmente em 2048. As outras duas cópias terão endereços iniciais 4096 e 6144, respectivamente. Os laços seguintes obtêm nas linhas **DATA** os padrões que serão modificados. As cópias da tabela original são necessárias, porque apenas as letras e os números são redefinidos, excluindo-se, portanto, os demais símbolos.

As linhas 100 a 180 criam as letras futuristas na tabela que começa em 4096, enquanto as linhas 200 a 280 produzem as letras de fôrma, utilizando a tabela iniciada em 6144. Essas linhas se parecem muito com as primeiras do programa responsável pela mudança do conjunto de caracteres.

As telas secundárias são obtidas pelas linhas 300 a 420. A variável **J** determina o endereço inicial da tela produzida pela sub-rotina da linha 360. Assim, a primeira tela começando em 0 é criada pela linha 300.

A sub-rotina iniciada a partir da linha 360 preenche as diferentes telas com vários caracteres gráficos usando o comando **VPOKE** na linha 370. Como o código do caractere depende da variável **J**, ele resultará diferente a cada nova tela. Além disso, na linha 380 surge uma mensagem que identifica a tela através de seu endereço inicial. O laço das linhas 390 a 410, usando o comando **VPOKE**, imprime a mensagem. Esta se destaca por meio de um espaço claro criado pela linha 365.

A sub-rotina da linha 360 é chamada pela primeira vez na linha 300, para criar a tela original. A posição da tela pode ser mudada através da repetição da linha 320 para diferentes valores da variável **J**. Nessa linha, a variável **BASE(0)** e, portanto, o endereço inicial da Tabela de Nomes, assume o valor **J**. Para cada nova tela, a sub-rotina 360 é chamada outra vez pela linha 330. Depois de produzidas todas as telas secundárias, o programa prossegue na linha 420, onde os valores de **BASE(0)** e de **BASE(2)** são restituídos ao normal.

A porção do programa que abrange as linhas 430 a 510 torna possível a exibição instantânea de cada uma das telas criadas, escritas com qualquer dos três conjuntos de caracteres. Esta mudança é obtida pela atribuição de um novo valor a uma variável **BASE**. As linhas 440 e 450 permitem "avançar" dentro da VRAM através da tecla "seta para baixo". Novas telas vão sendo mostradas instantaneamente, à medida que se altera o valor de **BASE(0)**. Por outro lado, as linhas 460 e 470 possibilitam "retroceder" dentro da VRAM. São necessárias duas linhas para cada movimento, devido ao grande intervalo entre a tela original (endereço inicial 0) e as secundárias. Este espaço é ocupado pelas Tabelas de Padrões.

Com as linhas 480 e 490 podemos mudar instantaneamente o conjunto de caracteres, alterando o valor de **BASE(2)**, endereço inicial da Tabela de Padrões. Para isso, o programa detecta as teclas "seta para a direita" e "seta para a esquerda".

A linha 500 possibilita terminar o programa sem maiores problemas através da tecla **<ESC>**. Essa linha restitui **BASE(0)** e **BASE(2)** aos seus valores iniciais antes de encerrar.

Este programa poderia ser mais ilustrativo se preenchêssemos cada tela secundária com um texto, de modo que parecessem páginas de um livro. Trata-se, entretanto, de uma tarefa difícil em razão da enorme quantidade de linhas necessárias.

ALGUMAS LIMITAÇÕES

Interrompido o programa, ainda é possível obter uma mudança automática do conjunto de caracteres digitando:

```
BASE(2)=4096
```

ou

```
BASE(2)=6144
```

Para recuperar o conjunto original, digite o seguinte:

```
BASE(2)=2048
```

ou

```
SCREEN 0
```

Ao fazer isto, o usuário notará algo estranho no cursor: normalmente quando o sobrepomos a um caractere na tela, esse caractere se inverte. Por exemplo: se era claro sobre fundo escuro, torna-se escuro, sobre fundo claro. Quando mudamos a posição da Tabela de Padrões, entretanto, o cursor permanece como um espaço em branco, que apaga o caractere subjacente e desaparece na porção vazia da tela.

A inversão do caractere sob o cursor é realizada na Tabela de Padrões, pelo sistema operacional do MSX. Isto se dá através das posições 2040 a 2047 da VRAM. Quando mudamos a **BASE(2)**, o processo não é mais visível.

Comandos como **PRINT**, **TAB** e **LOCATE** só funcionam quando a **BASE(0)** tem valor 0, ou seja, só imprimem na Tabela de Nomes em sua posição original. Portanto, apenas por intermédio de **VPOKE** é possível imprimir nas telas secundárias; todas as mensagens de erro são impressas na Tabela de Nomes original, mesmo que uma tela secundária esteja sendo mostrada no vídeo.

Além de tudo isso, existem algumas limitações nos valores da instrução **BASE**. O endereço inicial de uma tabela de nomes — **BASE(0)** no caso de **SCREEN 0** — deverá ser múltiplo de 1024. Da mesma forma, o endereço inicial de uma tabela de padrões — **BASE(2)** em **SCREEN 0** — terá que ser múltiplo de 2048. Voltaremos a este assunto oportunamente, quando explicarmos os registros do chip de vídeo-VDP.

DESENHO ARQUITETÔNICO (1)

- O MÉTODO TRADICIONAL
- COMO USAR PAPEL E CANETA
- TIRE AS MEDIDAS
- DESENHOS E ESCALA
- O GIRO DAS PEÇAS

Deixe para o micro o trabalho "pesado" do planejamento de um ambiente. Com este projeto computadorizado, você pode distribuir seus móveis com o simples toque de algumas teclas.

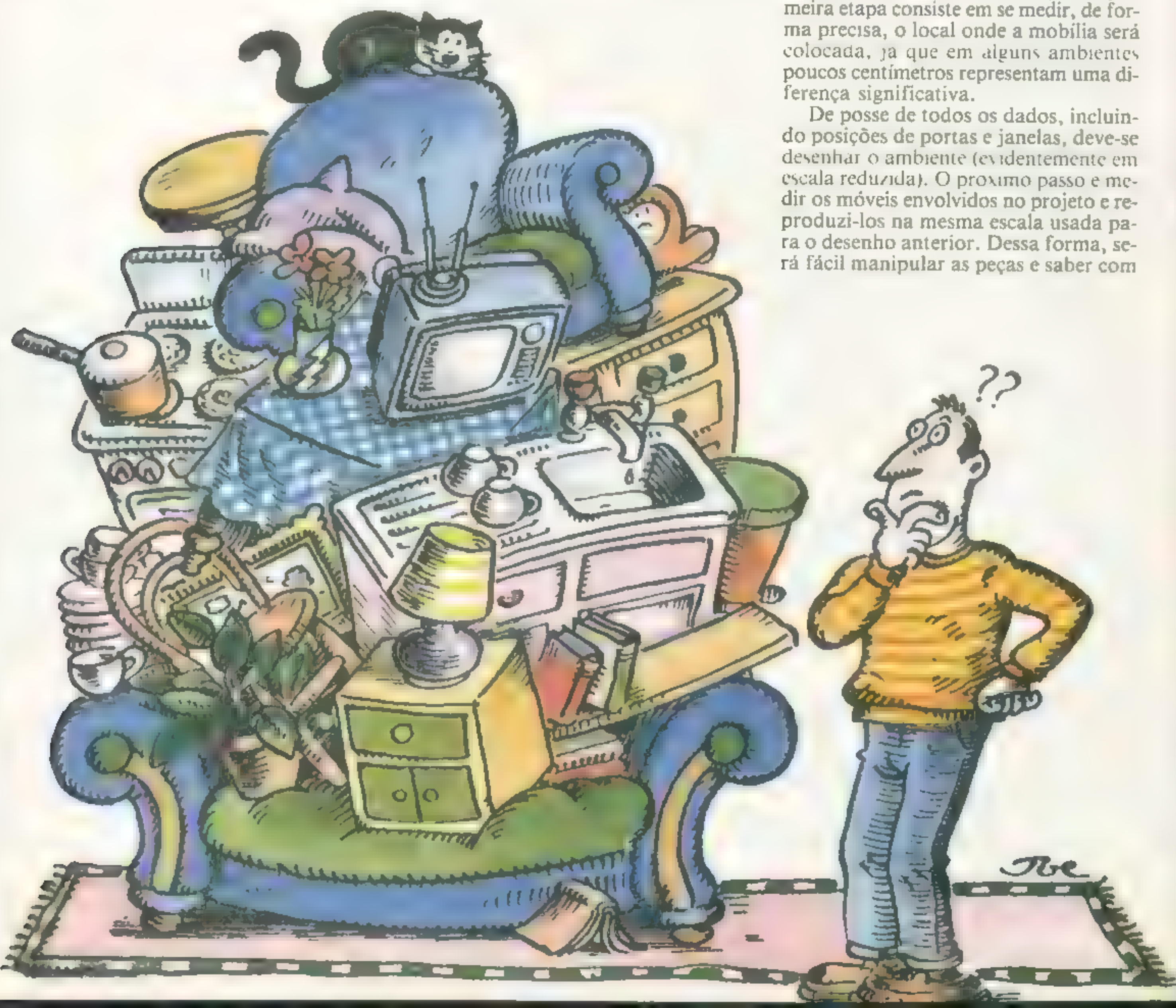
Conseguir uma melhor disposição dos móveis num ambiente nem sempre é algo fácil, e certamente é uma tarefa cansativa. Normalmente, optamos por arrastar a mobília de um lado para outro, até encontrarmos a melhor posição para as diferentes peças. O desânimo aparece quando constatamos que nem tudo se encaixa nos espaços disponíveis.

Um método mais simples consiste em fazer um planejamento prévio, desenhando cada móvel, com precisão, para em seguida adequá-los ao ambiente (obviamente é mais simples movimentar um retângulo que arrastar um sofá).

Apresentamos aqui, entretanto, uma terceira alternativa ainda mais fácil e moderna, utilizando seu microcomputador pessoal. O vídeo gráfico pode substituir o trabalho do papel e da caneta, com a grande vantagem de permitir fáceis correções. Além disso, os detalhes da mobília podem ser armazenados na memória do computador, eliminando a necessidade de redesenhar uma peça cada vez que quisermos movê-la.

O projeto computadorizado é semelhante ao projeto feito em papel. A primeira etapa consiste em se medir, de forma precisa, o local onde a mobília será colocada, já que em alguns ambientes poucos centímetros representam uma diferença significativa.

De posse de todos os dados, incluindo posições de portas e janelas, deve-se desenhar o ambiente (evidentemente em escala reduzida). O próximo passo é medir os móveis envolvidos no projeto e reproduzi-los na mesma escala usada para o desenho anterior. Dessa forma, será fácil manipular as peças e saber com



exatidão, por exemplo, qual a melhor posição para o piano na sala ou se a geladeira caberá em um determinado canto da cozinha.

Apresentamos aqui apenas metade do programa, que é composto de sete opções. O restante virá no próximo artigo.

As sete opções oferecidas no menu principal são as seguintes:

Opção 1 - Reproduz o ambiente: tomando por base sua maior medida, desenha-o em escala que caiba na tela do micro. As dimensões devem ser indicadas em metros, junto com suas respectivas posições (**C**ima, **B**aixo, **E**squerda, **D**ireita). Na representação das paredes, consideraremos duas distâncias e duas direções, permitindo, assim, o desenho na diagonal. Na opção 1, especificamos também a posição das portas e janelas.

Opção 2 - Possibilita a movimentação das peças da mobília já definidas no programa. São móveis de cozinha — talvez o ambiente mais difícil de ser planejado — e incluem armario, fogão, máquina de lavar louça, pia e geladeira, todos com medidas padronizadas. Qualquer alteração desejada deverá ser feita através da opção 3. O mesmo desenho de uma peça poderá ser colocado em diferentes lugares. Dessa forma, é possível, por exemplo, dispor quatro cadeiras iguais ao redor de uma mesa, posicionando a mesma definiçao da peça nos lugares escolhidos. Tudo que temos a fazer é selecionar o móvel acionando algumas teclas.

Opção 3 - Permite redefinir as peças já contidas no programa e incluir, no máximo, outras cinco, desde que não tenham mais que dez lados — o que é suficiente para desenhar qualquer móvel. As novas peças serão desenhadas automaticamente, tomando como base a escala da reprodução do ambiente.

Opção 4 - Permite salvar nosso projeto (opção um) e seu conteudo (opçoes 2 e 3) em disco ou fita.

Opçao 5 - Carrega um projeto e seu conteúdo de uma fita ou disco.

Opção 6 - Cuida da impressão. Como apenas o Spectrum é capaz de imprimir a cópia do projeto diretamente, esta opção ficará incompleta para os demais micros. Ela pode ser usada para chamar uma rotina que despeje o conteúdo da tela na impressora ou um outro programa em BASIC que faça a mesma tarefa. Em um próximo artigo, aprenderemos como criar essa rotina e obter cópias impressas de nossos projetos.

Opção 7 - E incumbida de terminar e sair do programa.

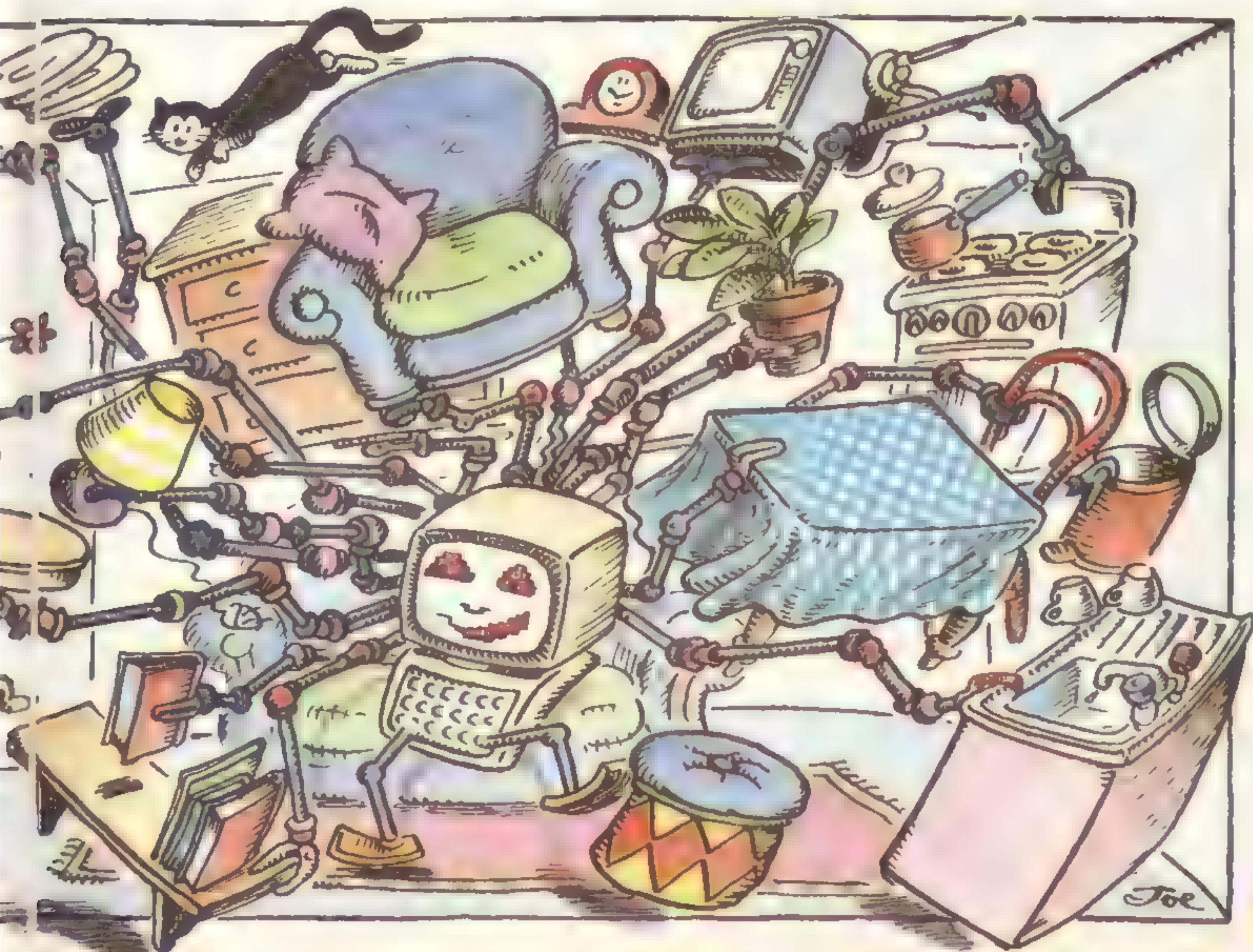
A segunda parte deste programa trará uma explicação mais detalhada de cada opção, além de conter a outra metade da listagem do programa. Mas atenção: é necessário digitar as duas partes para que o programa rode com segurança; portanto, grave a primeira antes de digitar a segunda.



```
10 PCLEAR 8:CLEAR 2000
20 DEF FNA(XM)=1.9*SC*XM
30 DIM O$(9),S(10)
40 O$(0)="DD100;DC50;DE100;DB50
;BD8;BC4;DD40;DC34;DE40;DB34;"
50 O$(1)="DD50;DC60,DE50;DB60;B
D10;BC10;DD10;DC10;DE10,DB10;BD
20;DD10,DC10;DE10;DB10,BC20;DC1
0;DD10;DB10;DE10;BE10;DE10;DC10
;DD10;DB10;BE20;BC15,DD50;"
60 O$(2)="DD100;DC60;DE100;DB60
."
70 O$(3)="DD30;DC30;DE30;DB30;D
C20;DD30;"
80 O$(4)="DD60;DC60,DE60;DB60;"
90 CLS
100 PRINT @96,TAB(6)"1: PLANEJA
R AMBIENTE"
110 PRINT TAB(6)"2: DESENHAR LA
YOUT"
120 PRINT TAB(6)"3. DESENHAR MO
BILIA"
130 PRINT TAB(6)"4: GRAVAR PROJ
ETO"
140 PRINT TAB(6)"5· CARREGAR PR
OJETO"
150 PRINT TAB(6)"6  IMPRIMIR PR
OJETO"
160 PRINT TAB(6)"7: SAIDA"
170 PRINT @422,"FACA A OPCAO";:
INPUT N
180 IF N<1 OR N>7 THEN 90
190 IF N=2 AND F1=0 THEN CLS:PR
INT"VOCE DEVE PRIMEIRO SELECION
AR   OPCAO 1":SOUND 1,20:GOTO 9
0
200 IF N=1 THEN F1=1
210 ON N GOTO 350,1120,830,1700
,1750,1790,230
220 GOTO 90
230 CLS·PCLS·END
240 RF=0:COLOR 0
250 LINE(200,0)-(255,191),PSET,
B
260 DRAW"BM206,10;S4;A0,NR4D6R4
BR2U6D3R4D3U6BR2D6R4U6NL4BR2D6R
4U6NL4BR2NR4D3R4D3NL4BR2NR4U3NR
2U3R4BR4BD2DBD2D"
270 IF RF=1 THEN RETURN
280 DRAW"BM203,30;D6R2NU3R2U6BR
6R4D6L4U6BR10D6R4U3L4R3U3L3BR10
NR4D3R4D3NL4BR2U6R4D3L4BR6U3NR4
D6R4"
290 DRAW"BM218,43;D6U3NR2U3R4BF
6NR4D6R4"
300 RETURN
310 RF=1:COLOR 0,1:LINE(201,1)-
(254,190),PRESET,BF.GOSUB 250
320 DRAW"BM208,30;ND6R4D3L4BR10
BU3D6R3E1U4H1L3BR10;ND6R4D3L4R1
F3BR6U6R4L4D3R2"
330 RETURN
340 DRAW"BM213,70;D4F1R2E1U4H1L
2G1BU1BR8D3R3BR5U3R4D3NL4D3BR8
UBU2U2R2U2L4D2":RETURN
350 CLS
360 PRINT"COMPRIMENTO MAXIMO DO
 COMODO(M)"
370 INPUT LE
380 IF LE>100 OR LE<3 THEN 350
390 SC=100/LE
400 PMODE 4,1:COLOR 0,1:PCLS:SC
REEN 1,0
410 XM=0:YM=LE
420 GOSUB 240
430 XX=FNA(XM):YY=FNA(YM)·IF PP
OINT(XX,YY)=1 THEN PSET(XX,YY,0
) ELSE PSET(XX,YY,1)
440 I$=INKEY$:IF I$="" THEN 430
450 OX=XM:OY=YM
460 IF I$=" " THEN COLOR 0:GOTO
 530
470 IF I$="B" THEN COLOR 1·GOTO
 530
480 IF I$="C" THEN 400
490 IF I$="F" THEN FOR K=1 TO 4
```



```
:PCOPY K TO K+4:NEXT:GOTO 90
500 IF I$="W" THEN 710
510 IF I$="O" THEN 800
520 GOTO 430
530 CLS
540 FOR A=1 TO 2
550 PRINT"DIRECAO";A;" C/B/E/D
";·INPUT D$(A)
560 IF D$(A)="" [illegible]
570 PRINT "DISTANCIA",A, INPUT
D(A)
580 IF INSTR(1."CBED".D$(A))=0
THEN 550
590 NEXT
600 SCREEN 1.0:FOR A=1 TO 2
610 IF D$(A)="" THEN 670
620 IF D$(A)="E" THEN XM=XM-D(A
)
630 IF D$(A)="D" THEN XM=XM+D(A
)
640 IF D$(A)="C" THEN YM=YM-D(A
)
650 IF D$(A)="B" THEN YM=YM+D(A
)
660 NEXT A
670 IF XM<0 OR XM>LE OR YM<0 OR
 Y>LE THEN SOUND 1,2:XM=OX:YM=O
Y:GOTO 430
680 X1=FNA(OX):Y1=FNA(OY):X2=FN
A(XM):Y2=FNA(YM)
690 LINE(X1,Y1)-(X2,Y2),PSET
700 GOTO 430
710 CLS:INPUT"DIRECAO C/B/E/D";
D$
720 INPUT "DISTANCIA";D
730 X1=FNA(OX):Y1=FNA(OY)
740 POKE 178,2
750 IF D$="E" THEN XM=XM-D:X2=F
NA(XM):LINE(X1,Y1)-(X2,Y1+3),PS
ET,BF
760 IF D$="D" THEN XM=XM+D:X2=F
NA(XM):LINE(X1,Y1)-(X2,Y1+3),PS
ET,BF
770 IF D$="C" THEN YM=YM-D:Y2=F
NA(YM):LINE(X1,Y1)-(X1+3,Y1),PS
ET,BF
780 IF D$="B" THEN YM=YM+D:Y2=F
NA(YM):LINE(X1,Y1)-(X1+3,Y2),PS
ET,BF
790 SCREEN 1,0:GOTO 430
800 CLS:INPUT"DIRECAO C/B/E/D";
D$(1)
810 INPUT"DISTANCIA";D(1)
820 D$(2)="":COLOR 1:GOTO 600
830 CLS
840 INPUT"DEFINIR ITEM NUMERO
(0-9)";N
850 IF N<0 OR N>9 THEN 830
860 O$(N)=""
870 PRINT"USE <ESPACO> PARA LIN
HA":PRINT"USE 'B' PARA LINHA VA
ZIA":FOR D=1 TO 1000:NEXT
880 PMODE 4,1:COLOR 0,1:PCLS:SC
REEN 1,0
890 X=75·Y=145
900 DRAW"BM75,150;R100NG3NH3;BM
70,145,U100NF3NG3"
910 IF PPOINT(X.Y)=0 THEN PSET(
X,Y,1) ELSE PSET (X,Y,0)
920 I$=INKEY$:IF I$="" THEN 910
930 OX=X:OY=Y
940 IF I$="C" THEN 830
950 IF I$="F" THEN 90
960 IF I$=" " THEN COLOR 0
970 IF I$<>" " THENCOLOR 1
```



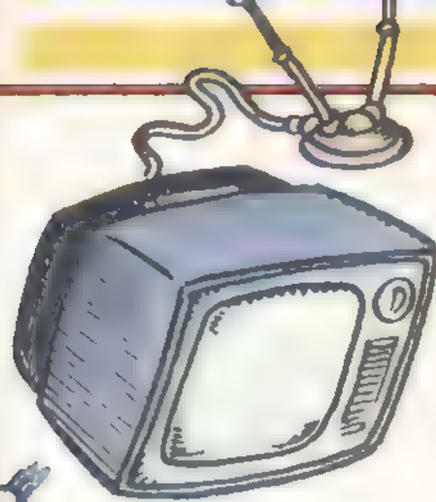
```
980 IF I$=" " THEN O$(N)=O$(N)+
"D" ELSE O$(N)=O$(N)+"B"
990 CLS:INPUT "DIRECAO C/B/E/D"
,D$
1000 IF INSTR(1,"CBED",D$)=0 TH
EN 990
1010 INPUT"DISTANCIA (CM)";D
1020 IF D<=0 OR D>200 THEN 1010
1030 SCREEN 1,0
1040 IF D$="E" THEN X=X-D/2
1050 IF D$="D" THEN X=X+D/2
1060 IF D$="C" THEN Y=Y-D/2
1070 IF D$="B" THEN Y=Y+D/2
1080 IF X<75 OR X>175 OR Y<45 O
R Y>145 THEN SOUND 1,3:X=OX:Y=O
Y:GOTO 910
1090 LINE(OX,OY)-(X,Y),PSET
1100 O$(N)=O$(N)+D$+STR$(D)+";"
1110 GOTO 910
1120 FOR K=1 TO 4:PCOPY K+4 TO
K:NEXT:CLS
1130 PMODE 4,1:SCREEN 1,0
1140 GOSUB 310
1150 X=128:Y=96:RT=0
1160 IF X<3 THEN X=3
1170 IF Y<3 THEN Y=3
1180 GET(X-3,Y-3)-(X+3,Y+3),S,G
1190 DRAW"C0;BM"+STR$(X)+","+ST
R$(Y)+"NU3ND3NL3NR3"
1200 I$=INKEY$:IF I$="" THEN 12
00
```



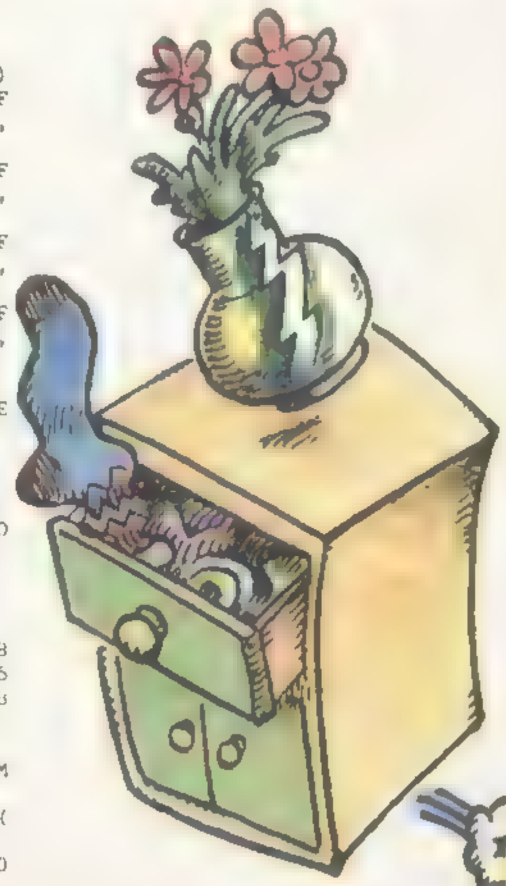
```
5 POKE 23658,8
10 BORDER 0: PAPER 0: INK 4:
CLS
12 GOSUB 8000
20 GOSUB 7000: PRINT INVERSE
1;AT 2,23;"[1]AMB";AT 3,23;"[
2]PLAN";AT 4,23;"[3]MOB";AT 5
,23;"[4]GRAV";AT 6,23;"[5]CAR
R";AT 7,23;"[6]IMPR";AT 8,23;
"[7]SAIDA"
30 LET K$="1234567": GOSUB
7040: GOSUB 7000: IF Z=55
THEN STOP
40 GOSUB 1000*(Z-49)+1000:
GOTO 20
1000 LET NF=1: GOSUB 6080
1005 PLOT 0,0: LET X=0: LET Y=0
1010 PRINT PAPER 2; INK 6;AT 2
1,22;"MAXIMO ";MAX
1015 PRINT INVERSE 1;AT 2,22;"
[J]JANELA";AT 3,22;"[P]PORTA";A
T 4,22;"[B]BRANCO";AT 5,22;"[ ]
DESENHA";AT 6,22;"[S]SAIDA";AT
7,22;""
1016 LET K$="JPBS ": GOSUB 7040
: INK 3*(Z=74)+2*(Z=80)+7*(Z=32
)
1020 IF Z=83 THEN INK 4: RETUR
N
1025 GOSUB 7010: LET DX=X+D*((D
$="D")*SC)-D*((D$="E")*SC): LET
DY=Y+D*((D$="C")*SC)-D*((D$="B
")*SC): GOSUB 7010
1030 LET DX=DX+D*((D$="D")*SC)-
D*((D$="E")*SC): LET DY=DY+D*((
D$="C")*SC)-D*((D$="B")*SC)
1032 IF DX>175 OR DX<0 OR DY>17
5 OR DY<0 THEN PRINT FLASH 1;
AT 7,23;"ERRO": PAUSE 100: GOTO
1015
1035 DRAW DX-X,DY-Y: LET X=PEEK
23677: LET Y=PEEK 23678: LET X
$=STR$ (X/SC): LET Y$=STR$ (Y/S
C)
1037 IF LEN X$<3 THEN LET X$=F
$( TO 3-LEN X$)+X$
1038 IF LEN Y$<3 THEN LET Y$=F
$( TO 3-LEN Y$)+Y$
1039 LET X$=X$( TO 3): LET Y$=Y
$( TO 3): PRINT INK 7;AT 10,24
;"DX=";X$;AT 11,24;"DY=";Y$
1040 GOTO 1016
2000 IF NF=0 THEN RETURN
2005 FOR N=1 TO 10: LET Q$="["+
STR$ N+"]"+O$(N): PRINT INVERS
E 1;AT N*2,24+4-LEN Q$;Q$: NEXT
N
2010 LET R=0: GOSUB 6060: LET C
U=2
2020 PRINT INVERSE 1;AT CU,29;
CHR$ 144: FOR N=1 TO 10: NEXT N
: PRINT INVERSE 1;AT CU,29;" "
2030 LET K$=INKEY$: LET CU=CU-2
*(((K$="7")*(CU>2)))+2*(((K$="6
")*(CU<20)))
2040 IF K$<>"S" THEN GOTO 2020
2050 LET OB=CU/2: LET OX=85: LE
T OY=85
2057 LET F=1: GOSUB 6010
2060 GOSUB 7000: PRINT AT 5,24;
FLASH 1;"OBJ=";O$(OB); FLASH 0
;AT 7,22; INVERSE 1,"[5-8]MOVE"
;AT 8,22,"[C]COLOCA",AT 9,22,"[
H]HORARIO",AT 10,22,"[A]ANTHORA
";AT 11,22;"[S]SAIDA"
2070 LET K$="5678CAHS": GOSUB 7
040: IF Z=83 THEN LET R=0: LET
F=1: GOSUB 6010: GOTO 60
2080 LET F=1: GOSUB 6010: LET O
X=(X+2*((Z=56)*(OX<175))-2*((Z=
53)*(OX>1)): LET OY=OY+2*((Z=55
)*(OY<175))-2*((Z=54)*(OY>0))
2085 IF Z=72 OR Z=65 THEN GOSU
B 6020
2090 GOSUB 6010
2100 IF Z=67 THEN LET F=0: GOS
UB 6010: GOSUB 7000: GOTO 2050
2110 GOTO 2070
3000 PRINT AT 2,24;"PROJETO"
3012 INPUT "ENTRE FIGURA A DEFI
NIR ",OB: IF OB<1 OR OB>10 THEN
GOTO 3012
3013 INPUT "DIGITE NUMERO DE LA
DOS (1-15) ? ";S(OB): IF S(OB)<
1 OR S(OB)>15 THEN GOTO 3013
3014 INPUT "CODIGO DE DUAS LETR
AS ?";O$(OB)
3016 LET R=0: GOSUB 6060: LET O
X=80: LET OY=80
3017 FOR S=1 TO S(OB)*2 STEP 2
3020 FOR N=1 TO 2: GOSUB 7010:
LET D=D/125: LET O(OB,S)=O(OB,S
)+((D*(-1*(D$="E")+(D$="D")))).
LET O(OB,S+1)=O(OB,S+1)+((D*(-
1*(D$="B")+(D$="C"))))
3030 NEXT N
3040 NEXT S
3050 LET F=1: GOSUB 6010: INPUT
"ESTA CORRETO (S/N) ?";S$: IF
S$="N" THEN GOSUB 6010: FOR N=
1 TO S(OB)*2+1: LET O(OB,N)=0:
NEXT N: GOTO 3012
3060 GOSUB 6010: RETURN
4000 GOSUB 6200
4015 IF Z=83 THEN SAVE E$SCREE
N$ : RETURN
4020 SAVE E$ DATA O()
4030 SAVE E$ DATA S()
4040 SAVE E$ DATA O$()
4050 RETURN
5000 GOSUB 6200
5010 IF Z=83 THEN LOAD E$SCREE
N$ : RETURN
5020 LOAD E$ DATA O()
5030 LOAD E$ DATA S()
5040 LOAD E$ DATA O$()
5050 RETURN
6000 COPY : RETURN
6010 OVER F: INK 7: PLOT OX,OY:
FOR N=1 TO (S(OB)*2) STEP 2: D
RAW O(OB,N)*CY-O(OB,N+1)*CX,O(O
B,N)*CX+O(OB,N+1)*CY: NEXT N
6014 OVER 0: INK 4: RETURN
6020 LET R=R+(2*(Z=65))-(2*(Z=7
2)): IF R>360 THEN LET R=0
6030 IF R<0 THEN LET R=360-R
6060 LET A=R*(PI/180): LET CY=S
C*COS A: LET CX=SC*SIN A: RETUR
N
6080 INPUT "ENTRE MAXIMA DIMENS
AO ?";MAX
6090 LET SC=175/MAX
6100 RETURN
6200 PRINT INVERSE 1;AT 10,23;
"[T]TELA";AT 11,23,"[D]DADOS":
LET K$="TD": GOSUB 7040: INPUT
"ENTRE NOME DO ARQUIVO";E$: RET
URN
7000 FOR N=0 TO 21: PRINT PAPE
R 4;AT N,22,"          ": NEXT
N: PRINT AT 0,25;"MENU": RETURN
```



```
5 SCREEN 2,0:COLOR 1,15,15
10 KEY OFF:GOTO 2000
20 DEF FNA(XM)=1.9*SC*XM
30 DIM O$(9),S(10)
40 O$(0)="DD100,DC50;DE100;DB50
;BD8;BC8;DD40;DC34;DE40;DB34;"
50 O$(1)="DD50;DC60;DE50;DB60;B
D10;BC10;DD10,DC10;DE10,DB10,BD
20,DD10;DC10;DE10;DB10;BC20;DC1
0;DD10;DB10;DE10,BE10;DE10,DC10
;DD10,DB10,BE20;BC15;DD50;"
60 O$(2)="DD100;DC60;DE100;DB60
;"
70 O$(3)="DD30,DC30,DE30;DB30;D
C20;DD30;"
80 O$(4)="DD60;DC60;DE60,DB60;"
85 CLS
90 SCREEN 0
100 LOCATE 8,5:PRINT"1: PLANEJA
R AMBIENTE"
110 LOCATE 8,6:PRINT"2: DESENHA
R LAYOUT"
120 LOCATE 8,7:PRINT"3: DESENHA
R MOBILIA"
130 LOCATE 8,8:PRINT"4: GRAVAR
PROJETO"
140 LOCATE 8,9:PRINT"5: CARREGA
R PROJETO"
150 LOCATE 8,10:PRINT"6: IMPRIM
IR PROJETO"
160 LOCATE 8,11:PRINT"7: TERMIN
```



```
AR"
170 LOCATE 8,14:INPUT"SUA ESCOL
HA ";N
180 IF N<1 OR N>7 THEN 170
190 IF N=2 AND F1=0 THEN LOCATE
 5,18:PRINT"<escolha primeiro a
 opção 1>":GOTO 170
200 IF N=1 THEN F1=1
210 ON N GOTO 350,1120,830,1700
,1750,1790,230
220 GOTO 90
230 CLS:END
240 RF=0
250 LINE(200,0)-(255,191),1,B
260 DRAW"BM206,10;S4;A0;NR4D6R4
BR2U6D3R4D3U6BR2D6R4U6NL4BR2D6R
4U6NL4BR2NR4D3R4D3NL4BR2NR4U3NR
2U3R4BR4BD2DBD2D"
270 IF RF=1 THEN RETURN
280 DRAW"BM203,30;D6R2NU3R2U6BR
6R4D6L4U6BR10D6R4U3L4R3U3L3BR10
NR4D3R4D3NL4BR2U6R4D3L4BR6U3NR4
D6R4"
290 DRAW"BM218,43;D6U3NR2U3R4BR
6NR4D6R4"
300 RETURN
310 RF=1:LINE(201,1)-(254,190),
15,BF:GOSUB 250
320 DRAW"BM208,30;ND6R4D3L4BR10
BU3D6R3E1U4H1L3BR10;ND6R4D3L4R1
F3BR6U6R4L4D3R2"
330 RETURN
340 DRAW"BM213,70;D4F1R2E1U4H1L
2G1BU1BR8BD3R3BR5U3R4D3NL4D3BR8
UBU2U2R2U2L4D2":RETURN
350 CLS
360 PRINT"COMPRIMENTO MAXIMO DO
 AMBIENTE"
370 INPUT"(em metros)";LE
380 IF LE<3 OR LE>100 THEN 350
390 SC=100/LE
400 SCREEN 2
410 XM=0:YM=LE
420 GOSUB 240:A=USR(0)
430 XX=FNA(XM):YY=FNA(YM):IF PO
INT(XX,YY)<>1 THEN PSET(XX,YY),
1
440 I$=INKEY$:IF I$="" THEN 430
450 OX=XM:OY=YM
455 D$(1)="":D$(2)=""
460 IF I$=" " THEN CL=1:GOTO 530
470 IF I$="B" THEN CL=15:GOTO 5
30
480 IF I$="C" THEN 400
490 IF I$="F" THEN A=USR(0):GOT
O90
500 IF I$="W" THEN 710
510 IF I$="O" THEN CL=15:GOTO 8
00
520 GOTO 430
530 SCREEN 0
540 FOR A=1 TO 2
550 PRINT"Direção";A;" C/B/D/E"
,:INPUT D$(A)
560 IF D$(A)="" THEN 600
570 PRINT"Distância";A;" (metro
s)"; INPUT D(A)
580 IF INSTR(1,"CBDE",D$(A))=0
THEN550
590 NEXT
600 SCREEN 2:A=USR2(0):A=USR1(0
)
605 FOR A=1 TO 2
610 IF D$(A)="" THEN 670
620 IF D$(A)="E" THEN XM=XM-D(A
)
630 IF D$(A)="D" THEN XM=XM+D(A
)
640 IF D$(A)="C" THEN YM=YM-D(A
)
650 IF D$(A)="B" THEN YM=YM+D(A
)
660 NEXT A
670 IF XM<0 OR XM>LE OR YM<0 OR
 YM>LE THEN BEEP:XM=OX:YM=OY:GO
TO 430
680 X1=FNA(OX):Y1=FNA(OY):X2=FN
A(XM):Y2=FNA(YM)
690 LINE(X1,Y1)-(X2,Y2),CL
695 A=USR(0)
700 GOTO 430
710 SCREEN 0:INPUT"Direção   C/
B/D/E";D$
720 INPUT"Distância ";D
730 X1=FNA(OX):Y1=FNA(OY)
735 SCREEN2:A=USR2(0):A=USR1(0)
750 IF D$="E" THEN XM=XM-D:X2=F
NA(XM):LINE(X1,Y1+1)-(X2,Y1+1),
1:LINE(X1,Y1)-(X2,Y2),1
760 IF D$="D" THEN XM=XM+D:X2=F
NA(XM):LINE(X1,Y1+1)-(X2,Y1+1),
1:LINE(X1,Y1)-(X2,Y2),1
770 IF D$="C" THEN YM=YM-D:Y2=F
NA(YM):LINE(X1+1,Y1)-(X1+1,Y2),
1:LINE(X1,Y1)-(X2,Y2),1
780 IF D$="B" THEN YM=YM+D:Y2=F
NA(YM):LINE(X1+1,Y1)-(X1+1,Y2),
1:LINE(X1,Y1)-(X2,Y2),1
790 A=USR(0):GOTO 430
800 CLS:INPUT"Direção   C/B/D/E
";D$(1)
810 INPUT"Distancia ";D(1)
820 D$(2)="":GOTO 600
830 CLS
840 INPUT"DEFINIR A PEÇA NUMERO
 (0-9)";N
850 IF N<0 OR N>9 THEN 830
860 O$(N)=""
870 SCREEN 2:RF=1:GOSUB 250
880 DRAW"BM218,43;NR4D3R4D3NL4B
R2U6R4D3L4BR6U3NR4D6R4;BM210,56
;D6R4U3L4R3U3L3BR15 6U3NR2U3R4B
R10NR4D6R4"
890 X=75:Y=145
900 DRAW"BM75,150,R100NG3NH3,BM
70,145,U100NF3NG3"
910 IF POINT(X,Y)<>1 THEN PSET(
X,Y),1
920 I$=INKEY$:IF I$="" THEN 920
925 A=USR3(0)
930 OX=X:OY=Y
940 IF I$="C" THEN 830
950 IF I$="F" THEN 90
960 IF I$=" " THEN CL=1
970 IF I$<>" " THEN CL=15
980 IF I$=" " THEN O$(N)=O$(N)+
"D" ELSE O$(N)=O$(N)+"B"
990 SCREEN0:INPUT"Direção   C/B
/D/E",D$
1000 IF INSTR(1,"CBDE",D$)=0 TH
EN 990
1010 INPUT"Distância (cms)",D
1020 IF D<=0 OR D>200 THEN 1010
1030 SCREEN2:A=USR2(0):A=USR4(0
)
1040 IF D$="E" THEN X=X-D/2
1050 IF D$="D" THEN X=X+D/2
1060 IF D$="C" THEN Y=Y-D/2
1070 IF D$="B" THEN Y=Y+D/2
1080 IF X<75 OR X>175 OR Y<45 O
R Y>145 THEN BEEP:X=OX:Y=OY:GOT
O 910
1090 LINE(OX,OY)-(X,Y),CL
1100 O$(N)=O$(N)+D$+MID$(STR$(D
),2)+","
1110 GOTO 910
1120 SCREEN 2
1130 A=USR2(0):A=USR1(0)
1140 GOSUB 310
1150 X=128:Y=96:RT=0
1160 IF X<3 THEN X=3
1170 IF Y<3 THEN Y=3
1190 PUTSPRITE0,(X,Y),1,0
1200 I$=INKEY$:IF I$="" THEN 120
0
```



A MATEMÁTICA DA IRREGULARIDADE (2)

Neste artigo, você verá como aplicar o que aprendeu sobre dimensões fracionadas na simulação de imagens da natureza, como uma montanha ou um floco de neve.

No artigo da página 1356, vimos como obter imagens fascinantes por meio de programas recursivos muito simples. Essas imagens, também denominadas *figuras fractais*, são geradas matematicamente na fronteira entre o que chamamos de regular e irregular.

Com a técnica fractal podemos conseguir formas muito mais proximas das observadas na natureza que com os modelos construídos a partir da ciência tradicional. Mostraremos aqui como fazer simulações desse tipo no seu micro.

O primeiro programa desenha uma das formas mais simétricas da natureza: a forma hexagonal de um floco de neve.

```
10 BORDER 0· PAPER 0· INK 5:
BRIGHT 1: CLS
20 LET AN=2*ATN (1)/3: LET S2
=2/SQR (3)
30 LET XC·127  LET YC=90: LET
S=120: LET C=2
50 GOSUB 1000
60 STOP
1000 LET S=S/3  IF S<1 THEN  LE
T S=S*3· RETURN
1020 PLOT  INVERSE 1, OVER 1;IN
T (XC+S2*S*SIN (-AN)),(YC-S2*S*
COS (-AN))· FOR K=0 TO 8*ATN (1
)-AN STEP 2*AN
1030 DRAW XC+2*S*SIN (K)-PEEK 2
3677,YC-2*S*COS (K) PEEK 23678
1040 DRAW XC+S2*S*SIN (K+AN) PE
EK 23677,YC-S2*S*COS (K+AN) PEE
K 23678
1050 NEXT K
1060 LET C=C-1: GOSUB 1000
1070 LET YC YC 1 36*S· GOSUB 10
00
1080 LET YC=YC+.68*S: LET XC=XC
+1 19*S· GOSUB 1000
1090 LET YC=YC+1.36*S  GOSUB 10
00
1100 LET YC=YC+ 68*S· LET XC=XC
-1.19*S· GOSUB 1000
1110 LET YC=YC-.68*S· LET XC=XC
·1 19*S· GOSUB 1000
1120 LET YC=YC-1.36*S: GOSUB 10
00
1130 LET YC=YC+ 68*S· LET XC=XC
+1 19*S· LET S S*3· LET C=C+1·
RETURN
```

```
10 PMODE 3,1·PCLS:SCREEN 1,0
20 AN=2*ATN(1)/3:S2=2/SQR(3)
30 XC=127:YC=95:S=135:C=4
50 GOSUB 1000
60 GOTO 60
1000 S=S/3:IF S<1 THEN S=S*3:RE
TURN
1010 IF C=2 THEN COLOR 4 ELSE I
F C=1 THEN COLOR 2 ELSE COLOR C
1020 DRAW"BM"+STR$(INT(XC+S2*S*
SIN(-AN)))+","+STR$(INT(YC-S2*S
*COS(-AN))):FOR K=0 TO 8*ATN(1)
-AN STEP 2*AN
1030 LINE -(XC+2*S*SIN(K),YC-2*
S*COS(K)),PSET
1040 LINE -(XC+S2*S*SIN(K+AN),Y
C-S2*S*COS(K+AN)),PSET
1050 NEXT:PAINT(XC,YC)
1060 C=C 1 GOSUB 1000
1070 YC=YC-1.36*S:GOSUB 1000
1080 YC=YC+.68*S:XC=XC+1.19*S:G
OSUB 1000
1090 YC=YC+1.36*S:GOSUB 1000
1100 YC=YC+.68*S XC=XC-1.19*S G
OSUB 1000
1110 YC=YC  68*S.XC=XC-1.19*S:G
OSUB 1000
1120 YC=YC-1.36*S·GOSUB 1000
1130 YC=YC+.68*S.XC=XC+1.19*S:S
=S*3·C C+1·RETURN
```



```
10  HGR2
20 AN = 2 *  ATN (1) / 3:S2 = 2
 /  SQR (3)
30 XC = 127:YC = 95:S = 135·C =
 4
50  GOSUB 1000
60  GOTO 60
1000 S = S / 3· IF S < 1 THEN S
 = S * 3· RETURN
1020  HPLOT  INT (XC + S2 * S *
 SIN ( - AN)), INT (YC - S2 *
S *  COS ( - AN)): FOR K = 0 TO
 8 *  ATN (1) - AN STEP 2 * AN
1030  HPLOT  TO XC + 2 * S *  S
IN (K),YC - 2 * S *  COS (K)
1040  HPLOT  TO XC + S2 * S *
SIN (K + AN),YC - S2 * S *  COS
 (K + AN)
1050  NEXT
1060 C = C - 1: GOSUB 1000
1070 YC = YC - 1.36 * S: GOSUB
1000
1080 YC   YC + .68 * S:XC = XC
+ 1.19 * S: GOSUB 1000
1090 YC = YC + 1.36 * S: GOSUB
1000
1100 YC = YC + .68 * S:XC = XC
- 1.19 * S: GOSUB 1000
1110 YC = YC - .68 * S:XC = XC
- 1.19 * S: GOSUB 1000
1120 YC = YC - 1.36 * S: GOSUB
1000
1130 YC = YC + .68 * S:XC = XC
+ 1.19 * S:S = S * 3:C = C + 1:
 RETURN
```



```
10 SCREEN 2
20 AN=2*ATN(1)/3·S2=2/SQR(3)
30 XC=127·YC=95·S=135·C=4
50 GOSUB 1000
60 GOTO 60
1000 S=S/3:IF S<1 THEN S=S*3:RE
TURN
1010 IF C=2 THEN COLOR 8 ELSE I
F C=1 THEN COLOR 11 ELSE COLOR
C
1020 DRAW "BM"+STR$(INT(XC+S2*S
*SIN(-AN)))+","+STR$(INT(YC-S2*
S*COS(-AN)))·FOR K=0 TO 8*ATN(1
)-AN STEP 2*AN
1030 LINE (XC+2*S*SIN(K),YC-2*S
*COS(K))
1040 LINE-(XC+S2*S*SIN(K+AN),YC
-S2*S*COS(K+AN))
1050 NEXT·PAINT(XC,YC)
1060 C=C-1·GOSUB 1000
1070 YC YC 1 36*S·GOSUB 1000
1080 YC YC+ 68*S·XC XC+1 19*S·G
OSUB 1000
1090 YC YC+1 36*S·GOSUB 1000
1100 YC=YC+.68*S·XC=XC-1.19*S:G
OSUB 1000
1110 YC YC- 68*S·XC XC 1 19*S·G
OSUB 1000
1120 YC=YC 1.36*S·GOSUB 1000
1130 YC YC+ 68*S XC XC+1 19*S·S
=S*3:C=C+1:RETURN
```

Esse programa é baseado na *curva floco de neve*, originalmente desenhada por Von Koch. A figura criada se assemelha a um cristal de gelo em um floco de neve ou a uma ilha com litoral muito recortado. A linha 20 especifica um triângulo eqüilátero (seus ângulos medem 60 graus e os lados são iguais) com um fator de escala definido em **S2**. A linha 30 define as primeiras coordenadas X e Y do centro, um fator de escala inicial e a cor do primeiro desenho (exceto no Apple). A linha 50 chama a sub-rotina que desenha a estrela de seis pontas que compõe a figura.

SIMETRIA E ASSIMETRIA

Apesar da irregularidade do desenho, a simetria está presente na forma da estrela. Como ocorre com frequência na

- MODELOS DE SIMETRIA
- PROGRAMA DO FLOCO DE NEVE
- FORMAS ASSIMETRICAS
- UMA MONTANHA NO MICRO
- GERADOR DE FIGURAS



natureza, a ordem e o caos se combinam nessa figura. Porém, muitas estruturas simuladas pela técnica fractal são totalmente assimétricas. É o caso das curvas de um rio, das crateras da Lua, das veias e artérias do corpo humano e do contorno das montanhas. O que distingue essas estruturas das formas simétricas geradas matematicamente são os elementos aleatórios que entram em sua composição. Podemos simular tais elementos no computador por meio da função **RND**. O programa que se segue gera a imagem de uma montanha:

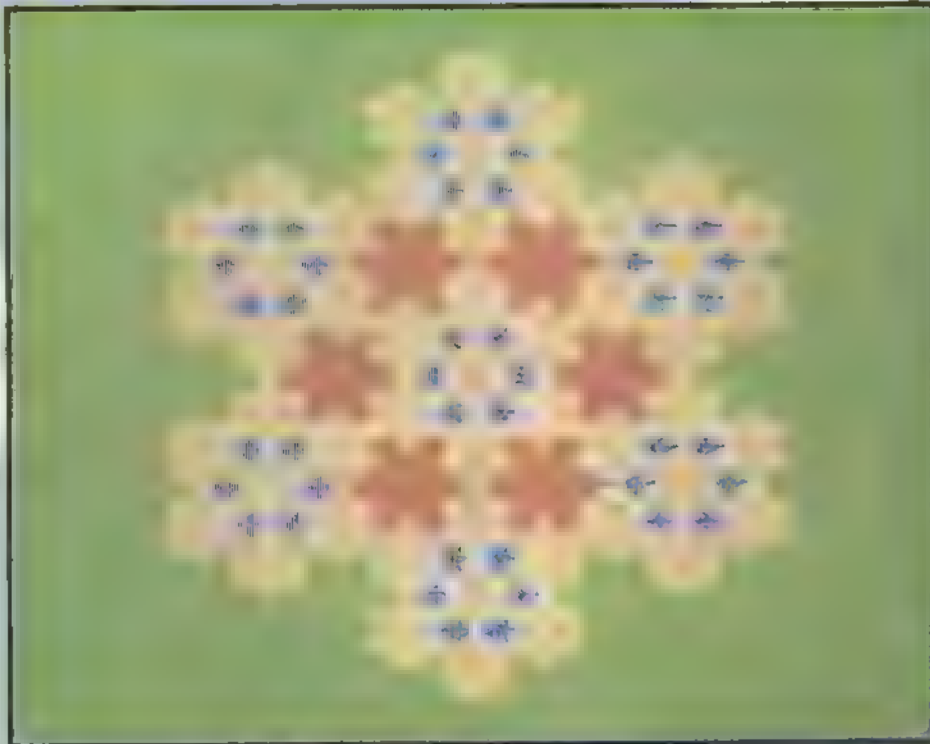
S

```
10 BORDER 0: PAPER 0: INK 7:
BRIGHT 1: CLS
```

A técnica dos fractais é responsável pelas mais perfeitas imagens de formas naturais simuladas no computador. A construção de uma imagem complexa como a escarpa nevada desta página só é possível em computadores gráficos de maior porte. Porém, a mesma técnica pode ser utilizada em micros domésticos, com resultado semelhante ao mostrado na ilustração do alto.

```
15 PRINT AT 6,2; INVERSE 1;"
GERADOR DE MONTANHA FRACTAL "
20 DIM C(200,2,2): LET F=1·
LET G=2: LET C(1,1,2)=25: LET
C(1,1,1)=0
22 INPUT "DIGITE 'RESOLUCAO'
DA MONTANHA  [16-100] ? ";S
23 IF S<16 OR S>100 THEN
GOTO 22
24 INPUT "DIGITE GRAU DE 'RUG
OSIDADE'[1-5]?";RG
25 IF RG<1 OR RG>5 THEN  GOTO
 24
26 DEF FN R(X)=RG-((RND*X)*(2
*RG))
27 PAPER 1: CLS
30 LET L=230/S· LET H=L/(SQR
3)
40 FOR K=1 TO S+1  LET C(K,1,
1)=C(1,1,1)+L*K-FN R(1): LET
C(K,1,2)=C(K-1,1,2)-FN R(1):
NEXT K
50 FOR J=1 TO S· FOR K=1 TO S
-J+1
60 LET C(K,G,1)=FN R(1)+(C(K,
F,1)+C(K+1,F,1))/2
70 LET C(K,G,2)=FN R(1)+H+(C(
K,F,2)+C(K+1,F,2))/2
80 PLOT C(K,F,1),C(K,F,2):
DRAW C(K+1,F,1)-PEEK 23677,C(
K+1,F,2)-PEEK 23678
90 DRAW C(K,G,1)-PEEK 23677,C
(K,G,2)-PEEK 23678· DRAW C(K,
F,1)-PEEK 23677,C(K,F,2)-PEEK
 23678
100 NEXT K: LET F=3-F: LET G=3
-F: NEXT J
110 FOR y=40 TO 0 STEP -.75
120 PLOT 0,y
130 FOR n=1 TO 100
140 LET a=RND*10
150 LET b=5 RND*10
160 IF a+PEEK 23677>255 THEN
LET n=100· DRAW 255-PEEK 23677
,b: GOTO 190
170 IF (PEEK 23678)+b<0 THEN
GOTO 150
180 DRAW a,b
190 NEXT n
200 NEXT y
210 FOR m=USR "a" TO USR "a"+7
· READ a: POKE m,a: NEXT m
220 DATA 16,56,84,16,56,84,146
,16
230 FOR n=1 TO 80
240 PRINT AT 17+INT (RND*4),
RND*31; BRIGHT 1; PAPER 4;
INVERSE 1;CHR$ 144;
250 NEXT n
260 PRINT #1: INVERSE 1;AT 0,4
;" RES=";S;"   RUGOSIDADE=";RG
;" "
270 GOTO 270
```

```
10 PMODE 4,1:PCLS:SCREEN 1,1
20 DIM C(200,1,1):F=0:G=1·C(0,0
,1)=150 C(0,0,0)=10
30 S=80:L=230/S:H=L/SQR(3):DEFF
NR(X)=3-RND(0)*6
40 FOR K=1 TO S:C(K,0,0)=C(0,0,
0)+L*K-FNR(0):C(K,0,1)=C(K-1,0,
1)-FNR(0):NEXT
50 FOR J=1 TO S:FOR K=0 TO S-J
60 C(K,G,0)=FNR(0)+(C(K,F,0)+C(
K+1,F,0))/2
70 C(K,G,1)=FNR(0)-H+(C(K,F,1)+
C(K+1,F,1))/2
80 LINE (C(K,F,0),C(K,F,1))-(C(
K+1,F,0),C(K+1,F,1)),PSET
90 LINE -(C(K,G,0),C(K,G,1)),PS
ET:LINE-(C(K,F,0),C(K,F,1)),PSE
T
100 NEXT:F=1-F:G=1-F:NEXT
110 GOTO 110
```



```
10   HGR2
20   DIM C(200,1,1):F = 0:G = 1:
C(0,0,1) = 150:C(0,0,0) = 10
30 S = 80:L = 230 / S:H = L /
SQR (3): DEF  FN R(X) = 3 -  RN
D (1) * 6
40   FOR K = 1 TO S:C(K,0,0) = C
(0,0,0) + L * K -  FN R(0):C(K,
0,1) = C(K - 1,0,1) -  FN R(0):
 NEXT
50   FOR J = 1 TO S: FOR K = 0 T
O S - J
60 C(K,G,0) =  FN R(0) + (C(K,F
,0) + C(K + 1,F,0)) / 2
70 C(K,G,1) =  FN R(0) - H + (C
(K,F,1) + C(K + 1,F,1)) / 2
80   HPLOT C(K,F,0),C(K,F,1) TO
C(K + 1,F,0),C(K + 1,F,1)
90   HPLOT  TO C(K,G,0),C(K,G,1)
: HPLOT  TO C(K,F,0),C(K,F,1)
100   NEXT :F = 1 - F:G = 1 - F:
 NEXT
110   GOTO 110
```

```
10 SCREEN 2
20 DIM C(200,1,1)·F=0·G=1:C(0,0
,1)=150:C(0,0,0)=10
30 S=80:L=230/S:H=L/SQR(3):DEFF
NR(X)=3-RND(1)*6
40 FOR K=1 TO S:C(K,0,0)=C(0,0,
0)+L*K-FNR(0):C(K,0,1)=C(K-1,0,
1)-FNR(0) NEXT
50 FOR J=1 TO S:FOR K=0 TO S-J
60 C(K,G,0)=FNR(0)+(C(K,F,0)+C(
K+1,F,0))/2
70 C(K,G,1)=FNR(0)-H+(C(K,F,1)+
C(K+1,F,1))/2
80 LINE(C(K,F,0),C(K,F,1))-(C(K
+1,F,0),C(K+1,F,1))
90 LINE-(C(K,G,0),C(K,G,1)):LIN
E-(C(K,F,0),C(K,F,1))
100 NEXT·F=1-F:G=1-F:NEXT
110 GOTO 110
```

A linha 30 define o fator de escala dos triângulos e especifica o comprimento e a altura de um lado. A linha 40 preenche duas matrizes com as coordenadas iniciais de cada triângulo. Observe que há um fator aleatório — portanto, os valores irão variar em um pequeno intervalo a cada execução do programa. A linha 50 inicializa dois laços: um para desenhar os triângulos horizontalmente na tela, e outro para colocá-los mais acima.

O vértice de cada triângulo é especificado nas linhas 60 (coordenada X) e 70 (coordenada Y). A linha 80 move o cursor para o canto esquerdo do triân-

gulo e desenha sua base. No programa para o Spectrum, os **PEEK** subtraídos das coordenadas asseguram que os pontos não caiam fora da tela, o que provocaria uma interrupção do programa ao se tentar executar o **DRAW.** Na linha 90, o **DRAW** é direcionado para o vértice do triângulo e depois para o canto esquerdo (o início). A linha 100 completa o primeiro laço. Este desenha uma fileira de triângulos e, após redefinir algumas variáveis, de maneira a deslocar a fileira um pouco mais para cima, recomeça a construção da montanha. Na versão para o Spectrum, o programa completa a figura desenhando "árvores" na base da montanha.

FORMAS VARIADAS

Modificando os valores de algumas variáveis, você poderá obter muitas imagens diferentes. Mas, lembre-se, essa variação atinge apenas o tamanho, a posição e outros detalhes, mas não a forma geral da figura.

O próximo programa permite que você introduza uma forma inicial, construindo, a partir dela, uma figura fractal. O grande número de sub-rotinas recursivas pode ser um problema no Apple — esse micro só admite que uma sub-rotina seja chamada por ela mesma 24 vezes. Por isso deve-se ter cuidado ao escolher o nível de recursão.

S

```
10 POKE 23658,8: LET F=0: LET
A$="": LET CL=1
20 DIM X(100): DIM Y(100):
DIM T(30): DIM U(30): DIM V(
60): DIM W(60): DIM J(100)
25 BORDER 0: PAPER 7: INK 0:
CLS
30 GOSUB 140
40 GOSUB 350
50 INPUT "No DE NIVEIS DE REC
URSAO ";NR: IF ABS (INT (NR))
<1 THEN  GOTO 50
60 LET F=1  LET N=0: CLS
70 PLOT  INVERSE 1: OVER 1;
INT (127+X(P)),INT (85+Y(P))
80 GOSUB 500: IF P>=2 THEN
GOTO 80
90 LET A$=INKEY$  IF A$=""
THEN  GOTO 90
100 PRINT #1;"S PARA SAIR, QUA
LQUER OUTRA P/  CONTINUAR"
110 LET A$=INKEY$: IF A$=""
THEN  GOTO 110
120 IF A$<>"S" THEN  GOTO 25
130 CLS : STOP
140 IF F=0 THEN  GOTO 170
150 PRINT "MESMA FORMA INICIAL
(S/N) ? ";
160 LET A$=INKEY$  IF A$<>"S"
AND A$<>"N" THEN  GOTO 160
165 PRINT A$
170 IF F=0 OR A$="N" THEN
GOSUB 230
180 FOR K=2 TO CV+1
190 LET P=K 1
200 LET X(P)=T(K): LET Y(P)=U(
K)
210 NEXT K
220 RETURN
230 INPUT "No DE VERTICES ";VT
240 FOR L=2 TO VT+1
250 INPUT "VERTICE ";(L 1);"="
;X,Y
260 LET T(L)=X*85  LET U(L)=y*
85
270 IF L=2 THEN  PLOT INT (127
+T(L)),INT (85+U(L))
275 IF L<>2 THEN  DRAW 127+T(L
) PEEK 23677,85+U(L) PEEK
23678
280 NEXT L
290 PRINT "CURVA FECHADA (S/N)
? ";
300 LET A$=INKEY$  IF A$<>"N"
AND A$<>"S" THEN  GOTO 300
305 PRINT A$
310 IF A$="N" THEN  LET CV=VT:
PAUSE 0: RETURN
320 LET CV=VT+1: LET T(CV+1)=T
(2): LET U(CV+1) U(2)
330 DRAW 127+T(CV) PEEK 23677,
85+U(CV) PEEK 23678
340 RETURN
350 CLS   IF F=0 THEN  GOTO
380
360 PRINT "MESMO GERADOR (S/N)
? ";
370 LET A$=INKEY$: IF A$<>"S"
AND A$   "N" THEN  GOTO 370
375 PRINT A$
380 IF F=0 OR A$="N" THEN
GOSUB 400
390 RETURN
400 INPUT "No DE VERTICES DO G
ERADOR NAO  IN LUINDO AS EXTR
EMIDADES (0,0  E  ,0) " GN
420 PLOT  INVERSE 1, OVER 1,85
,85
430 FOR M=2 TO GN+1
440 INPUT "VERTICE ",(M-1);"="
;X,Y
450 IF ABS (INT (X),,1 OR ABS
(INT (Y))>1 THEN  GOTO 44
460 LET V(M) X  LET w(M) Y
LET X X*85+85  LET Y=85+Y*85:
DRAW X PEEK 23677,Y-PEEK 23678
470 NEXT M
480 DRAW 175-PEEK 23677,85
PEEK 23678: PAUSE 0
490 RETURN
500 IF NR N THEN  GOSUB 520
505 IF NR<>N THEN  GOSUB 570
510 RETURN
520 FOR W 1 TO GN+1
530 LET P=P 1
540 IF ABS X(P)>127 OR ABS Y(P
)>85 THEN  GOTO 560
550 DRAW 127+X(P) PEEK 23677,
85+Y(P) PEEK 23678
560 NEXT W  RETURN
570 LET N N+1
580 IF N=1 THEN  LET JM=CV 1
585 IF N<>1 THEN  LET JM=GN+1
590 FOR E=1 TO JM
595 IF P  THEN  LET E JM
NEXT E: RETURN
600 LET TX=X(P): LET TY=Y(P)
610 LET BX X(P 1)  LET BY Y(P
1)
620 LET DX=TX-BX: LET DY=TY-BY
630 FOR F 2 TO GN+1
640 LET X(P) DX*V F  DY*w F)+
BX
650 LET Y(P)=DY*V(F +DX*w F)+
BY
660 LET P=P+1
670 NEXT F
680 LET X(P) TX  LET Y P  TY
690 LET J CL) F   ET C  CL+1
GOSUB 500: LET CL=CL-1: LET E=
J(CL)
700 NEXT E
710 LET N N 1
720 RETURN
```

T

```
10 DIM X(50),Y(50),XT(50),YT(10
),XG(20),YG(20),J(50)
20 PMODE 4,1 COLOR 0,1 PCLS CLS
30 GOSUB 140
40 GOSUB 350
50 INPUT"NO DE NIVEIS DE RECURS
AO  ";NR NR=INT(NR) IF NR<1 THE
N 50
60 F=1 N=0 PCLS SCREEN 1,0
70 LINE  (127+X(P),96 Y(P)) PRE
SET
80 GOSUB 500:IF P>0 THEN 80
90 A$=INKEY$:IF A$="" THEN 90
100 CLS:PRINT"S PARA SAIR, QUAL
QUER OUTRA     TECLA PARA  ONTI
NUAR"
110 A$=INKEY$ IF A$="" THEN 110
120 IF A$ <>"S" THEN 20
130 CLS END
140 IF F 0 THEN 170
150 PRINT"MESMA FORMA INICIAL (
S/N)?";
160 A$=INKEY$:IF A$<>"S" AND A$
<>"N" THEN 160 ELSE PRINT AT A$
170 IF F=0 OR A$ "N" GOSUB 230
180 FOR K=1 TO CV
190 P=K 1
200 X(P)=XT(K):Y(P)=YT(K)
210 NEXT
220 RETURN
230 INPUT"NO DE VERTICES NA FOR
ME INICIAL ";VT:VT=INT(VT) IF V
T<1 THEN 230
240 FOR L=1 TO VT
250 PRINT"VERTICE ";L;:INPUT" -
";X,Y:IF ABS(X)>1 OR ABS(Y)>1
THEN 250
260 XT(L)=X*95 YT(L)=Y*95
270 IF L=1 THEN LINE  (127+XT(L
),96 YT(L)),PRESET ELSE LINE -(
127+XT(L),96 YT(L)),PSET
280 NEXT
290 PRINT"CURVA FECHADA  S/N)?";
300 A$ INKEY$ IF A$<>"N" AND A$
<>"S" THEN 300 ELSE PRINT A$
310 IF A$="N" THENCV VT GOSUB 7
30 RETURN
320 CV=VT+1:XT(CV)=XT(1):YT(CV)
=YT(1)
```

```
330 LINE-(127+XT(CV),96-YT(CV)),PSET:GOSUB 730
340 RETURN
350 PCLS:IF F=0 THEN 380
360 PRINT"MESMO GERADOR (S/N) ?";
370 A$=INKEY$:IF A$<>"S" AND A$<>"N" THEN 370 ELSE PRINT A$
380 IF F=0 OR A$="N" GOSUB 400
390 RETURN
400 PRINT"NO.DE VERTICES NO GERADOR"
410 INPUT"NAO INCLUINDO AS EXTREMIDADES (0,0) E (1,0) - ";GN:GN=INT(GN):IF GN<1 THEN 410
420 DRAW"BM80,96"
430 FOR M=1 TO GN
440 PRINT "VERTICE ";M;"DO GERADOR ";:INPUT "- ";X,Y
450 IF ABS(X)>1 OR ABS(Y)>1 THEN 440
460 XG(M)=X:YG(M)=Y:X=X*95+80:Y=96-Y*95:LINE-(X,Y),PSET
470 NEXT
480 LINE-(175,96),PSET:GOSUB 730
490 RETURN
500 IF NR=N GOSUB 520 ELSE GOSUB 570
510 RETURN
520 FOR L=1 TO GN+1
530 P=P-1
540 IF ABS(X(P))>127 OR ABS(Y(P))>95 THEN 560
550 LINE-(127+X(P),96-Y(P)),PSET
560 NEXT:RETURN
570 N=N+1
580 IF N=1 THEN JM=CV-1 ELSE JM=GN+1
590 FOR J=1 TO JM
600 TX=X(P):TY=Y(P)
610 BX=X(P-1):BY=Y(P-1)
620 DX=TX-BX:DY=TY-BY
630 FOR E=1 TO GN
640 X(P)=DX*XG(E)-DY*YG(E)+BX
650 Y(P)=DY*XG(E)+DX*YG(E)+BY
660 P=P+1
670 NEXT
680 X(P)=TX:Y(P)=TY
690 J(CL)=J:CL=CL+1:GOSUB 500:CL=CL-1:J=J(CL)
700 NEXT J
710 N=N-1
720 RETURN
730 A$=INKEY$:SCREEN 1,0:K=1000
740 K=K-1:IF K>0 AND INKEY$="" THEN 740
750 RETURN
```



```
5 HOME
10 DIM X(50),Y(50),XT(10),YT(10),XG(20),YG(20),J(50)
30 GOSUB 140
40 GOSUB 350
50 TEXT : HOME : INPUT "QUANTOS NIVEIS DE RECURSAO ";NR:NR = INT (NR): IF NR < 1 THEN 50
60 F = 1:N = 0: HGR2
70 HCOLOR= 3: HPLOT 127 + X(P),96 - Y(P)
80 GOSUB 500: IF P > 0 THEN 80
90 GET A$
100 TEXT : HOME : PRINT "<S> PARA SAIR E QUALQUER OUTRA PARA CON-TINUAR"
110 GET A$
120 IF A$ < > "S" THEN 30
130 HOME : END
140 IF F = 0 THEN 170
150 TEXT : HOME : PRINT "MESMA FORMA INICIAL?(S/N)";
160 GET A$: IF A$ < > "S" AND A$ < > "N" THEN 160
165 PRINT A$
170 IF F = 0 OR A$ = "N" THEN GOSUB 230
180 FOR K = 1 TO CV
190 P = K - 1
200 X(P) = XT(K):Y(P) = YT(K)
210 NEXT
220 RETURN
230 INPUT "QTOS. VERTICES NA FORMA INICIAL ";VT:VT = INT (VT): IF VT < 1 THEN 230
240 FOR L = 1 TO VT
250 PRINT "VERTICE ";L;: INPUT " - ";X,Y: IF ABS (X) > 1 OR ABS (Y) > 1 THEN 250
260 XT(L) = X * 95:YT(L) = Y * 95
265 NEXT
270 PRINT "CURVA FECHADA (S/N)?";
280 GET A$: IF A$ < > "N" AND A$ < > "S" THEN 280
285 PRINT A$: FOR T = 1 TO 300: NEXT : HGR2
290 FOR L = 1 TO VT
300 IF L = 1 THEN HCOLOR= 3: HPLOT 127 + XT(L),96 - YT(L)
302 IF L < > 1 THEN HCOLOR= 3: HPLOT TO 127 + XT(L),96 - YT(L)
307 NEXT
310 IF A$ = "N" THEN CV = VT: GOSUB 730: RETURN
320 CV = VT + 1:XT(CV) = XT(1):YT(CV) = YT(1)
330 HPLOT TO 127 + XT(CV),96 - YT(CV): GOSUB 730
340 RETURN
350 IF F = 0 THEN 380
360 TEXT : HOME : PRINT "MESMO GERADOR?(S/N)";
370 GET A$: IF A$ < > "S" AND A$ < > "N" THEN 370
375 PRINT A$
380 IF F = 0 OR A$ = "N" THEN GOSUB 400
390 RETURN
400 TEXT : HOME : INPUT "QUANTOS VERTICES NO GERADOR NAO INCLUIN-DO AS EXTREMIDADES (0,0) E (1,0) ";GN
410 GN = INT (GN): IF GN < 1 THEN 400
430 FOR M = 1 TO GN
440 PRINT "VERTICE ";M;" DO GERADOR";: INPUT " - ";XG(M),YG(M)
450 IF ABS (XG(M)) > 1 OR ABS (YG(M)) > 1 THEN 440
455 NEXT
457 HGR2 : HCOLOR= 3: HPLOT 80,96
458 FOR M = 1 TO GN
460 X = XG(M) * 95 + 80:Y = 96 - YG(M) * 95: HPLOT TO X,Y
470 NEXT
480 HPLOT TO 175,96: GOSUB 730
490 RETURN
500 IF NR = N THEN GOSUB 520
505 IF NR < > N THEN GOSUB 570
510 RETURN
520 FOR L = 1 TO GN + 1
530 P = P - 1
540 IF ABS (X(P)) > 127 OR ABS (Y(P)) > 95 THEN 560
550 HPLOT TO 127 + X(P),96 - Y(P)
560 NEXT : RETURN
570 N = N + 1
580 IF N = 1 THEN JM = CV - 1
585 IF N < > 1 THEN JM = GN + 1
590 FOR J = 1 TO JM
600 TX = X(P):TY = Y(P)
610 BX = X(P - 1):BY = Y(P - 1)
620 DX = TX - BX:DY = TY - BY
630 FOR E = 1 TO GN
640 X(P) = DX * XG(E) - DY * YG(E) + BX
650 Y(P) = DY * XG(E) + DX * YG(E) + BY
660 P = P + 1
670 NEXT E
680 X(P) = TX:Y(P) = TY
690 J(CL) = J:CL = CL + 1: GOSUB 500:CL = CL - 1:J = J(CL)
700 NEXT J
710 N = N - 1
730 FOR T = 1 TO 1000: NEXT
740 RETURN
```



```
5 CLS
10 DIM X(50),Y(50),XT(10),YT(10),XG(20),YG(20),J(50)
20 SCREEN 0
30 GOSUB 140
40 GOSUB 350
50 SCREEN 0:INPUT "QUANTOS NIVEIS DE RECURSAO ";NR:NR=INT(NR):IF NR<1 THEN 50
60 F=1:N=0:SCREEN 2
70 LINE-(127+X(P),96-Y(P)),4
80 GOSUB 500:IF P>0 THEN 80
90 A$=INKEY$:IF A$="" THEN 90
100 SCREEN 0:PRINT"<S> PARA SAIR E QUALQUER OUTRA PARA CONTINUAR"
110 A$=INKEY$:IF A$="" THEN 110
120 IF A$<>"S" THEN 20
130 CLS:END
140 IF F=0 THEN 170
150 SCREEN 0:PRINT"MESMA FORMA INICIAL?(S/N)";
160 A$=INKEY$:IF A$<>"S" AND A$<>"N" THEN 160 ELSE PRINT A$
170 IF F=0 OR A$="N" THEN GOSUB 230
180 FOR K=1 TO CV
```


Imagens de um gerador de figuras na tela.

```
190 P=K-1
200 X(P) XT(K)·Y(P) YT(K)
210 NEXT
220 RETURN
230 INPUT "QTOS  VERTICES NA FO
RMA INICIAL";VT·VT=INT(VT) IF V
T<1 THEN 230
240 FOR L 1 TO VT
250 PRINT "VERTICE ";L::INPUT"
- ":X,Y·IF ABS(X)>1 OR ABS(Y)>1
 THEN 250
260 XT(L)=X*95·YT(L)=Y*95
265 NEXT
270 PRINT"CURVA FECHADA (S/N) ?
 ";
280 AS=INKEYS·IFAS<>"N" AND AS<
>"S" THEN 280 ELSE PRINT AS·FOR
 T=1 TO 300·NEXT
290 SCREEN 2 FOR L 1 TO VT
300 IF L 1 THEN LINE  (127+XT(L
),96 YT(L)),4 ELSE LINE -(127+X
T(L),96 YT(L)),11
305 NEXT
310 IF AS="N" THEN CV=VT:GOSUB
730·RETURN
320 CV=VT+1 XT(CV)=XT(1)·YT(CV)
=YT(1)
330 LINE  (127+XT(CV),96-YT(CV)
),11:GOSUB 730
340 RETURN
350 IF F=0 THEN 380
360 SCREEN 0·PRINT"MESMO GERADO
R?(S/N)";
370 AS=INKEYS·IF AS<>"S" AND AS
<>"N" THEN 370 ELSE PRINT AS
380 IF F=0 OR AS="N" THEN GOSUB
 400
390 RETURN
400 INPUT"QUANTOS VERTICES NO G
ERADOR NAO INCLUINDO AS EXTREMI
DADES (0,0) E (1,0)";GN
410 GN=INT(GN) IF GN<1 THEN 400
430 FOR M=1 TO GN
440 PRINT"VERTICE ";M;" DO GERA
DOR", INPUT" - ",XG(M),YG(M)
450 IF ABS(X)>1 OR ABS(Y)>1 THE
N 440
455 NEXT
457 SCREEN 2 PSET(80,96),11
458 FOR M=1 TO GN
460 X=XG(M)*95+80·Y=96-YG(M)*95
:LINE-(X,Y),11
470 NEXT
480 LINE-(175,96),11:GOSUB 730
490 RETURN
500 IF NR=N THEN GOSUB 520 ELSE
 GOSUB 570
510 RETURN
520 FOR L=1 TO GN+1
530 P=P 1
540 IF ABS(X(P))>127 OR ABS(Y(P
))>95 THEN 560
550 LINE-(127+X(P),96-Y(P)),11
560 NEXT:RETURN
570 N=N+1
580 IF N=1 THEN JM=CV-1 ELSE JM
=GN+1
590 FOR J=1 TO JM
600 TX=X(P)·TY=Y(P)
610 BX=X(P 1):BY=Y(P 1)
620 DX=TX-BX:DY=TY BY
630 FOR E=1 TO GN
640 X(P)=DX*XG(E)-DY*YG(E)+BX
650 Y(P)=DY*XG(E)+DX*YG(E)+BY
660 P=P+1
670 NEXT
680 X(P)=TX·Y(P)=TY
690 J(CL)=J·CL=CL+1:GOSUB 500·C
L CL-1·J J(CL)
700 NEXT J
710 N N 1
720 RETURN
730 AS INKEYS·K=1000
740 K K-1:IF K>0 AND INKEYS=""
THEN 740
750 RETURN
```

Quando você executa o programa a linha 230 pede o número de vértices do desenho inicial que irá gerar a figura fractal. Convém traçar essa figura previamente em uma folha de papel. Marque dois pontos representando o início e o fim da linha e una-os por meio de pequenas retas. Conte o número de cantos e forneça esse dado ao computador. Não exagere no número de retas: lembre-se que você terá que especificar as coordenadas de cada vértice (linha 250). Estas devem ter valores compreendidos entre 1 e 1.

O laço entre as linhas 240 e 280 (240 e 305 nas versões para o MSX e o Apple) permite que você introduza as coordenadas e desenhe a forma inicial, determinando inclusive se sua figura será fechada ou aberta (linha 290).

Em seguida (linhas 400 a 490), o computador solicita ao usuário a definição da figura que substituirá cada linha reta do desenho inicial — essa figura é geralmente chamada de gerador. Como você fez para a forma inicial, desenhe a figura e passe a informação para o computador.

Finalmente, você deverá especificar o número de níveis de recursão. Quando esse valor é digitado, a linha 80 chama a sub-rotina que verifica se o programa está rodando pela primeira vez. Em caso afirmativo, o programa é desviado para a rotina principal (linha 570 à 720), que desenha a figura fractal. Se o programa já estava sendo rodado, o computador dá ao usuário a chance de redefinir o gerador.

TESTE

Para experimentar o programa, digite o valor 3 para o número de vértices da forma inicial. Em seguida, introduza as coordenadas -0.5 e -0.2 para o vértice 1; 0 e 0.4 para o vértice 2; 0.5 e -0.2 para o vértice 3. Ao responder N à pergunta "CURVA FECHADA?", aparecerá na tela um triângulo sem base. Como foi feito para a forma inicial, entre 3 para o número de vértices do gerador e defina as coordenadas: 0.2 e 0 para o vértice 1; 0.4 e 0.8 para o vértice 2; 0.6 e 0 para o vértice 3. Esses dados definem a imagem de um triângulo sem base sobre uma linha. Por fim, introduza o valor 5 para o nível de recursão (3 para o Apple) e observe a geração da figura fractal.

BITS E BYTES EM BASIC

A manipulação dos bits individuais de um byte de memória por meio de comandos do BASIC parece uma tarefa impossível. Mas não é: aprenda os truques de programação necessários para isso.

O byte é a unidade basica de memória usada em microcomputadores. Os diversos comandos da linguagem BASIC tratam o byte como a unidade mínima de informação: os caracteres individuais de uma cadeia alfanumérica, por exemplo, podem ser isolados e processados individualmente com comandos do tipo **MID$**, **LEFT$**, **RIGHT$**, **LEFTS$**, **CHR$**, **ASC** etc. Cada caractere ocupa um byte. Os valores numéricos, por sua vez, em geral ocupam dois ou mais bytes, dependendo de sua precisão (sobre a armazenagem de números, veja o artigo da página 894). Usando os comandos **VARPTR**, **PEEK** e **POKE**, temos acesso individual aos bytes onde variáveis, vídeo, teclado etc. estão armazenados.

O BIT

Existe, porém, uma unidade de informação menor que o byte. Essa unidade é o bit (*binary digit*, ou dígito binário, em inglês), que corresponde aos dígitos 0 e 1 que todo computador digital usa como base de representação numérica. Cada byte tem oito bits, numerados de 0 (o bit menos significativo) a 7 (o bit mais significativo).

Muitas vezes, surge a necessidade de manipular diretamente os bits da memória. Para isso, convém utilizar uma rotina em linguagem de máquina, que tem comandos específicos para o acesso e a manipulação de bits individuais e é mais rápida que qualquer rotina em linguagem de alto nível.

Entretanto, poucos sabem que é perfeitamente possível realizar manipulações diretas de bits com os comandos já existentes no BASIC. Não é tão rapido nem tão direto quanto em linguagem de máquina, mas funciona bem.

Neste artigo, você aprenderá vários truques que permitem a manipulação de bits em quase todos os microcomputadores. Os comandos do BASIC mais relevantes para esse fim são o **CHR$**, o **ASC** (ou **CODE**, no ZX-81) e os operadores lógicos **AND**, **NOT** e **OR**.

Com as funções **CHR$** e **POKE**, podemos criar bytes com uma determinada configuração de bits. **CHR$** é utilizada no trabalho com variáveis simbólicas dentro de um programa em BASIC; **POKE** nos dá acesso direto às locações absolutas da memória RAM.

CHR$(1), por exemplo, define um byte com o bit menos significativo igualado a 1, e todos os outros igualados a 0: 0000001 em binário. **CHR$(2)** gera um byte com os dois bits menos significativos igualados a 1: 00000011. **CHR$(3)** define o byte 00000010 e assim por diante. Com o auxílio de uma tabela de correspondência entre números decimais de 0 a 255 e os binários equivalentes, pode-se produzir qualquer padrão de bits ligados e desligados.

Para converter o padrão de bits em um número decimal, devemos considerar cada bit como o índice de uma potência de 2 e somar o resultado. Suponhamos um byte com os bits 0, 3 e 4 ligados:

```
00001101
```

O número decimal correspondente é:

$$2^4 + 2^3 + 2^0 = 16 + 8 + 1 = 25$$

Eis algumas expressões úteis para ligar ou desligar só o bit B de um byte armazenado em um caractere **B$**:



Para ligar:

```
B$=CHR$(CODE(B$) OR 2**B)
```

Para desligar:

```
B$=CHR$(CODE(B$) AND NOT 2**B)
```



Para ligar:

```
B$=CHR$(ASC(B$) OR 2**B)
```

Para desligar:

```
B$=CHR$(ASC(B$) AND NOT 2**B)
```



Para ligar:

```
B$=CHR$(ASC(B$) OR 2↑B%)
```

Para desligar:

```
B$=CHR$(ASC(B$) AND NOT 2↑B%)
```



Para ligar:

```
B$=CHR$(ASC(B$) OR 2^B%)
```

Para desligar:

```
B$=CHR$(ASC(B$) AND NOT 2^B%)
```

Observe que, com exceção dos micros da linha Sinclair, que não têm variáveis inteiras, usamos a notação **B%** para indicar que esse valor deve ser inteiro. Isso ajuda a evitar erros de cálculo, caso **B** seja um bit real de precisão simples.

Para tornar o bit B de um byte igual a 1, usamos o operador lógico **OR**. Suponhamos que você queira igualar a 1 o bit número 7 (o mais significativo) de um byte igual a 00001011. Teremos, então, a operação, em decimal:

$$25 \text{ OR } 2^7 = 25 \text{ OR } 128$$

o que, em binário, dá:

```
00001011  OR
10000000
--------
10001011
```

Note que **OR** tem o efeito de operar individualmente sobre cada par de bits na mesma posição, nos operandos, de acordo com o seguinte esquema:

```
0 OR 0  =  0
1 OR 0  =  1
0 OR 1  =  1
1 OR 1  =  1
```

Para colocar 0 em um bit qualquer do byte **B$**, usamos a operação **AND NOT**. De fato:

```
0 AND NOT 0 = 0 AND 1 = 0
1 AND NOT 0 = 1 AND 1 = 1
0 AND NOT 1 = 0 AND 0 = 0
1 AND NOT 1 = 1 AND 0 = 0
```

pois a operação **NOT** transforma um bit 0 em 1 e vice versa, equivalendo, portanto, a uma *inversão* do bit.

Se você quisesse transformar o biná-

rio 10001011 em 00001011, igualando o bit 7 a 0, teria:

```
     10001011 AND
NOT  10000000
```

que é a mesma coisa que:

```
10001011 AND
01111111
---------
00001011
```

Pode ser que, em vez de trabalhar com bytes armazenados em caracteres literais, você queira manipular os bits individuais de um número inteiro — que, na maioria dos micros, é armazenado em dois bytes contíguos, em um total de dezesseis bits. Nesse caso, use as expressões apresentadas a seguir. Mas lembre-se de que alguns computadores, como o ZX-81, podem armazenar de forma diferente um número inteiro.



Para ligar:

```
I% = I% OR 2↑B%
```

Para desligar.

```
I% = I% AND NOT 2↑B%
```



Para ligar:

```
I% = I% OR 2^B%
```

Para desligar:

```
I% = I% AND NOT 2^B%
```

Essas expressões funcionam com um conjunto de dezesseis bits — ou seja, se quisermos ligar o bit 13 do byte I%, faremos a operação:

```
I% = I% OR 2 ^ 12
```

ou

```
I% = I% OR 8192%
```

A expressão **AND NOT** desliga o B-ésimo bit de I%.

Os micros ZX-81 e Spectrum não podem utilizar essas expressões, já que não têm variáveis inteiras. Todos os outros micros podem, pois trabalham com a versão microsoft do BASIC.

COMBINAÇÃO DE BITS

É possível ligar uma combinação de bits em uma variavel literal ou inteira. Para isso, usa-se uma expressão de soma dos expoentes de 2 correspondentes a cada bit a ser ligado, combinados por operações **OR**. Para ligar os bits 3 e 5 do byte em **T$**, faça:



```
T$ = CHR$((2**3) OR (2**5))
```



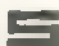
```
T$ = CHR$((2↑3) OR (2↑5))
```



```
T$ = CHR$((2^3) OR (2^5))
```

TESTANDO OS BITS

Da mesma forma que podemos ligar ou desligar bits individuais em linguagem BASIC, é bastante fácil examinar o valor desses bits, através de uma expressão simples:



```
R = CODE (B$) AND 2**B
```



```
R = ASC (B$) AND 2**B
```



```
R% = ASC(B$) AND 2↑B%
```

ou

```
R% = I% AND 2↑B%
```



```
R% = ASC(B$) AND 2^B%
```

ou

```
R% = I% AND I^B%
```

A operação **AND** isola apenas o valor do bit mascarado pelo byte igual a 2^B. Por exemplo: qual é o valor do bit número 6 do byte 01001011?

```
01001011  AND
01000000
---------
01000000
```

que equivale a 89 **AND** 64, cujo resultado é 64 em decimal.

Como o número resultante dessa operação é maior do que 0, o resultado é verdadeiro. Se fosse igual a 0, teríamos então um resultado falso.

Para transformar o resultado **R** em valor 1, fazemos:

```
R% = ABS(R%>(2 ↑ B% - 1))
```

Em nosso exemplo, isso daria:

```
ABS (64>63) = 1
```

Se R% fosse 0, teríamos:

```
ABS (0>63) = 0
```

Eis aqui um programa para testar todos os bits de um byte de entrada:



```
110 PRINT "ENTRE UM NUMERO EN-
TRE 0 E 255)
120 INPUT N
130 LET N$=CHR$(N)
140 FOR I=0 TO 7
150 PRINT "BIT ";I;" = ";
160 LET C=ASC(N$) AND 2**I
170 IF C THEN PRINT "SIM"
180 IF NOT C THEN PRINT "NAO"
190 NEXT I
```



O programa para o ZX-81 é o mesmo do Spectrum, com esta alteração:

```
160 LET C=CODE(N$) AND 2**I
```



```
110 PRINT "ENTRE UM NUMERO EN-
TRE 0 E 255)
120 INPUT N%
130 N$=CHR$(N%)
140 FOR I%=0 TO 7
150 PRINT "BIT ";I%;" = ";
160 C%=ASC(N$) AND 2↑I%
170 IF C% THEN PRINT "SIM"
180 IF NOT C% THEN PRINT "NAO"
190 NEXT I%
```



O programa para o MSX e o Apple é igual ao anterior, com a modificação:

```
160 C%=ASC(N$) AND 2^I%
```

MAPAS DE BITS

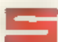
Até agora, vimos como manipular os bits de um único byte ou conjunto de dois bytes. Como uma cadeia alfanumérica contém até 255 caracteres, podemos manipular um conjunto muito maior de bits (um máximo de 8 × 255, ou 2040 bits). Esse conjunto é chamado de *cadeia de mapeamento de bits (bit-mapstring)*, e tem muitas aplicações em jogos, bancos de dados e outros programas de caráter profissional.

Por exemplo, é possível armazenar em uma cadeia desse tipo condições sim/não ou verdadeiro/falso. Se um bit em determinada posição dessa cadeia estiver ligado, temos uma condição sim, ou verdadeira; se estiver desligado, uma condição não, ou falsa.

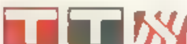
O comprimento da cadeia dependerá do número de condições (*flag bits* ou bits indicadores) que queremos incluir. Se o número de condições for **N**, o número de bytes da cadeia será:

```
L = INT(N/8)+1
```

Para inicializar um mapa **M$**, zeramos todos os bits:



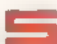
```
45 CLS
50 INPUT "NUMERO DE BITS NA CAD
EIA ";n
60 IF n>255 THEN GOTO 50
70 FOR i=1 TO INT(n/8)+1
80 LET m$=m$+CHR$(0)
90 NEXT i
```



```
45 CLS
50 INPUT "NUMERO DE BITS NA CAD
EIA ";N
60 IF N>255 THEN 50
70 M$=STRING$(INT(N/8)+1,0)
```

Para ligar, desligar e testar um bit qualquer do conjunto total de **N**, precisaremos de algumas funções poderosas: **FNL$**, **FND$** e **FNT$**.

Essas funções são bastante complexas, mas muito rápidas. Procure analisá-las passo a passo para entender o que elas executam:



```
20 DEF FNL$(B)=m$(TO INT(b/8))
+CHR$(ASC(m$(INT(b/8)+1)) OR 2
**(b-INT(b/8)*8))+m$(INT(b/8)+2
 TO)
30 DEF FND$(b)=m$(TO INT(b/8))
+CHR$(ASC(m$(INT(B/8)+1)) AND
NOT 2**(b-INT(b/8)*8))+m$(INT
b/8)+2 TO)
40 DEF FNT(b)=ABS((ASC(m$(INT
(b/8)+1)) AND 2**(b-INT(b/8)*
8))<>0)
```



```
20 DEF FNL$(B)=LEFT$(M$,INT(B/8
))+CHR$(ASC(MID$(M$,INT(B/8)+1,
1)) OR 2↑(B-INT(B/8)*8))+MID$(M
$,INT(B/8)+2)
30 DEF FND$(B)=LEFT$(M$,INT(B/8
))+CHR$(ASC(MID$(M$,INT(B/8)+1,
1)) AND NOT 2↑(B-INT(B/8)*8))+M
ID$(M$,INT(B/8)+2)
40 DEF FNT(B)=ABS((ASC(MID$(M$,
INT(B/8)+1)) AND 2↑(B-INT(B/8)*
8))<>0)
```

Por fim, acrescentamos o restante do programa de demonstração das funções de manipulação e consulta de bits:



```
200 PRINT "PROGRAMA DE DEMONSTR
ACAO DE MAPAS DE BITS"
210 INPUT "(L)IGA (D)ESLIGA (T)
ESTA (F)IM ";OP$
230 IF OP$="F" THEN STOP
240 INPUT "NUMERO DO BIT (0 A "
;N;")";B
260 IF B>N THEN GOTO 240
270 IF OP$="L" THEN M$=FNL$(B)
280 IF OP$="D" THEN M$=FND$(B)
290 IF OP$="T" THEN PRINT "BIT
";B," = ";FNT(B)
310 GOTO 210
```

APLICAÇÕES

Você não terá dificuldade em encontrar aplicações para os truques explicados neste artigo.



Qual é a vantagem de empregar cadeias alfanuméricas, em vez de conjuntos numéricos, para armazenar grupos de bits?

Como vimos no artigo *Armazenagem de Programas* (página 1001), a cadeia alfanumérica é armazenada na memória do microcomputador na forma de uma sequência contígua de bytes, cada qual ocupado por um caractere. O primeiro byte da sequência sempre indica o número de caracteres da variável literal (*string*). Esse número é "lido" por intermédio da função **LEN**, não havendo, no BASIC (como ocorre em outras linguagens), um byte encarregado de indicar ao interpretador onde termina o string.

A vantagem de empregar cadeias alfanuméricas decorre justamente desse sistema de armazenagem usado pelo interpretador BASIC.

A disposição sequencial dos bytes na memória facilita enormemente a programação de rotinas de acesso aos bits individuais de uma longa cadeia. Um string de trinta bytes, por exemplo, equivale a uma cadeia ininterrupta de 240 bits, que podem ser lidos ou modificados um a um, ou em grupos que se sobreponham aos limites entre bytes. Isso é muito mais difícil de ser feito quando a variável contém elementos separados, como é o caso dos conjuntos numéricos.

No desenvolvimento de jogos, por exemplo, a possibilidade de manipular bits amplia bastante as alternativas do programador. Entre outras coisas, você poderá modificar os gráficos impressos no vídeo alterando diretamente as locações de memória.

Os recursos de programação aqui examinados também serão úteis quando for necessário comprimir texto, em função de limitações da memória. Na série de artigos sobre o assunto, que iniciamos na página 1332, apresentamos diversos algoritmos interessantes para a redução do espaço ocupado por um texto em até 50%. A maioria desses algoritmos procura colocar dois ou mais códigos de caracteres em um único byte. Como exercício, tente implementar o supercompressor baseado na estatística de pares de letras em um texto (veja os artigos mencionados).

PROCESSADORES DE TEXTOS

- CORREÇÃO ELETRÔNICA
- LIMPEZA DO TEXTO NA TELA
- HARDWARE NECESSÁRIO
- MEMÓRIA
- IMPRESSÃO



Processamento de textos é uma solução elegante, quase mágica, que a tecnologia deu a muitos dos problemas que têm atribulado os escritores, desde a invenção da escrita.

É quase impossível produzir um texto perfeito logo na primeira tentativa. Em geral, cometemos erros ou, mais corriqueiramente, mudamos de idéia sobre o que desejamos transmitir. Como se sabe, a correção e modificação de textos já escritos toma tempo, e o resultado final acaba todo rabiscado. Pior, se houver muitas alterações, é preciso rebater ou reescrever todo o documento, e novos erros podem aparecer. Quem já experimentou escrever conhece bem esse torturante processo.

O processamento de textos por computador apresenta a vantagem imediata de facilitar a execução de alterações, tornando desnecessária a cansativa ta-

refa de "passar a limpo" o trabalho feito. Isso resulta em tremenda economia de tempo, o que é de utilidade mesmo para aqueles que não escrevem por necessidade ou profissão.

Já publicamos em *INPUT*, no artigo da página 576, um programa que coloca à disposição do usuário os recursos básicos de edição de texto. Se você tem interesse em adquirir um processador mais potente, convém conhecer mais detalhadamente suas possibilidades.

ADEUS, BORRACHA!

Processar um texto significa simplesmente digitá-lo no teclado de um computador e armazená-lo na memória. Portanto, usando um software adequado, você poderá fazer modificações e correções com rapidez e eficiência, e em seguida proceder à impressão. Se o texto for armazenado em fita ou disquete, será possível carregá-lo novamente, sempre que necessário, para outras modificações.

Como vimos no artigo já mencionado, com um programa elementar de processamento de texto, o usuário efetua as alterações "passeando" um cursor (geralmente um pequeno retângulo iluminado) sobre a tela, até a parte a ser modificada. Escreve então sobre o texto que ali se encontra. Este é apagado da memória e substituído pelo novo texto — sem rabiscos e sem complicações.

Inserir letras, palavras ou frases no meio de um parágrafo também é uma operação simples. Para realizá-la, recorre-se à função de inserção dos processadores de texto. Por meio dela, podem-se introduzir alguns espaços em branco e depois escrever sobre eles. O processo mais comum, porém, é simplesmente escrever o que se quer inserir, deixando por conta do programa rearranjar automaticamente o restante do texto, para dar lugar ao novo.

Estes são apenas dois dos numerosos recursos disponíveis em um processador de texto. A potência e sofisticação desse software ficam evidentes quando se consideram os diversos "truques eletrônicos" de que é capaz para auxiliar quem está escrevendo.

As funções de inserção e modificação de textos são chamadas *recursos de edição*. Entre estes incluem-se também a busca e a substituição de caracteres no texto. A função de busca permite a localização automática de uma seqüência específica de caracteres — uma palavra, ou mesmo uma frase — em uma parte ou em todo o texto. Suponhamos que, tendo digitado um texto muito longo, queiramos fazer uma modificação em certo parágrafo. Não nos lembramos onde ele está: sabemos apenas que contém as palavras *espiral inflacionária*. A função de busca nos ajudará a descobri-las rapidamente.

A função de substituição é mais poderosa ainda. Ela efetua a busca de uma seqüência determinada de caracteres por todo o texto, substituindo-a sempre que a encontrar por uma segunda seqüência. Isso é útil quando grafamos incorretamente uma palavra qualquer em vários pontos do texto, ou quando queremos trocar uma expressão empregada com freqüência por outra de significado mais preciso. Com a função de substituição, podemos também recorrer a abreviaturas ou códigos na digitação de um texto. Por exemplo, se vamos escrever diversas vezes *programa amistoso para o usuário* no texto, bastará digitar um asterisco, ou qualquer outro símbolo, em cada um dos pontos em que a frase deve aparecer, usando posteriormente a função de substituição.



HABILIDADES ESPECIAIS

Com o processador de texto, não é preciso ser um datilógrafo habilidoso para conseguir bons resultados. Mesmo que você só consiga datilografar com dois dedos, observará um aumento considerável na sua velocidade de trabalho, pois poderá avançar "a todo vapor", sem se preocupar com erros ou efeitos estéticos.

Se sua ortografia deixa a desejar, procure utilizar um processador de texto dotado de corretor ortográfico (*spell-checker*, em inglês). Ele percorre o seu texto "olhando" cada palavra, e comparando-a com um dicionário interno padrão (armazenado em disco). Palavras não reconhecidas — por terem erro ortográfico, ou, simplesmente, por não constarem do dicionário — são assinaladas. Neste ponto, dependendo do programa, você tem três opções: deixar a palavra indicada como está, corrigi-la manualmente, ou permitir que o programa a substitua pela versão correta (muitos programas mostram uma lista das palavras que poderiam substituir a versão assinalada no texto).

FORMATAÇÃO

Os processadores de texto mais potentes dispõem de uma série de recursos adicionais, que permitem executar o que chamamos de *formatação do texto*. A formatação tem por objetivo produzir um texto final arranjado segundo certo padrão estético, independente da manei-

ra como foi digitado. Os recursos de formatação usualmente permitem especificar o espaçamento entre linhas, o número de linhas por página, o número de toques por linha, o tamanho da folha onde vai ser impresso o texto, a numeração das páginas, cabeçalho etc. A função mais importante da formatação, porém (e a mais difícil de se realizar manualmente), é a justificação das margens — ou seja, a definição do alinhamento lateral das palavras. A seção de formatação dos processadores de texto realiza essa tarefa automaticamente. Muitos deles chegam à sofisticação de separar as sílabas das palavras, ao final de uma linha, conforme as regras do idioma.

Quanto à formatação, existem três tipos de processadores de texto. Todos eles se encarregam das mudanças de linha, permitindo ao usuário datilografar o texto de um parágrafo continuamente. Em geral, a tecla <ENTER> precisa ser pressionada apenas ao final de um parágrafo.

No primeiro tipo de processador de texto, quando uma palavra que está sendo datilografada não cabe na linha, o cursor passa para a linha seguinte do vídeo, sem se preocupar em dividi-la segundo as regras gramaticais. O texto fica difícil de ser lido na tela.



Nessa situação, o segundo tipo de processador de texto transporta a palavra inteira para a linha de baixo (método denominado *word-wrap*, em inglês). A leitura no vídeo torna-se bem mais fácil, mas a margem esquerda do texto fica irregular, o que no Brasil recebe o nome de *margem americana*.

O terceiro tipo de processador ajusta a margem esquerda na tela, de modo a proporcionar um alinhamento. Como já mencionamos, alguns sistemas chegam a *hifenar* automaticamente as palavras que se dividem ao final da linha, o que melhora consideravelmente a aparência do texto. Esse processador é denominado *WYSWYG* (do inglês *What You See What You Get*). Com ele, o usuário tem oportunidade de acompanhar na tela — à medida que vai escrevendo — a produção do texto final.

O tamanho do texto, evidentemente, não é limitado ao tamanho da tela. A maioria dos processadores de texto permite "rolar" um texto para cima ou para baixo, ou, ainda, para a esquerda ou para a direita, de modo que a tela funcione como uma "janela" que mostra parte do texto. Nos micros pessoais, a janela tem, em geral, vinte linhas por quarenta colunas.

REALCE O SEU TEXTO

Os processadores de texto oferecem diversas opções de realce de seções do texto, como negrito, texto inverso (caracteres brancos sobre fundo negro, na impressão), letras de tamanhos ou formatos diferentes (itálico, dupla largura etc.). Esses realces podem aparecer na tela e na impressora, ou apenas na impressora. Nesse caso, caracteres especiais de demarcação assinalam, na tela, o lugar onde se inicia e termina uma função de realce.

COMANDOS DE CONTROLE

Os programas mais simples de processamento são os *editores de texto*, com as funções básicas de inserção e modificação. Um editor se transforma em um processador de palavras com o acréscimo de diversas outras funções, que podem ser utilizadas pelo usuário no momento em que necessitar. A maneira de ativar essas funções varia de programa para programa, e é um elemento importante na *ergonomia* (facilidade de uso) do processador.

As diretivas de controle dos recursos



do programa são entradas pelo teclado, num processo tão fácil quanto o da entrada de texto. Os *comandos funcionais* consistem normalmente de teclas especiais, ou sequências de teclas. Entre as mais comuns, incluem-se as de retrocesso, as de controle dos cursores, <ENTER> etc. Bons comandos de edição são essenciais para facilitar o uso do programa. Em alguns casos, as seqüências de comando são muito complicadas e difíceis de memorizar — o que torna o programa pouco conveniente para o usuário ocasional.

Além desses, o programa possui comandos de controle de formatação cujo efeito só se observa, em geral, no momento da impressão. Nos programas WYSWYG, eles são "embutidos" no texto, sob a forma de caracteres invisíveis de controle. O vídeo responde a esses comandos de maneira bem semelhante ao que uma impressora responderia; isso possibilita que o usuário tenha uma visão prévia de como o documento ficará quando pronto.

Nos demais programas, os comandos de formatação são colocados explicitamente no texto, precedidos por sinais especiais. Por exemplo, um comando como:

```
.ME 20
```

introduzido no meio de um texto, indicaria ao processador que, naquele local, a margem esquerda deve ser modificada para vinte colunas.

CARTAS MÚLTIPLAS

Outro recurso de programas mais sofisticados de processamento de texto é a *mala direta* — a impressão de cartas múltiplas, endereçadas de forma personalizada a várias pessoas (veja o artigo da página 17).

Uma das maneiras mais simples de fazer isso consiste em escrever uma carta padronizada, com o auxílio do pro-

cessador de texto, e depois armazená-la em fita ou disco.

Para enviar essa carta a diferentes destinatários, bastará carregar o texto da mesma na memória e, utilizando as funções de edição, inserir nome, endereço e outros dados nos pontos apropriados, imprimindo-a a seguir.

Uma alternativa mais elaborada é o uso de um outro arquivo de textos — denominado *máscara* — contendo, por exemplo, o nome e o endereço de todas as pessoas para as quais se quer mandar uma carta. No texto da carta, por sua vez, introduzem-se caracteres especiais, que assinalam onde devem entrar os dados da máscara. Ao se fazer a impressão, o programa produzirá uma carta para cada destinatário, automaticamente, colocando os dados nos pontos certos do texto.

Um sistema de produção de cartas múltiplas tem variadas aplicações tanto domésticas (o envio de cartões de Natal personalizados, por exemplo), quanto profissionais (propaganda de um produto ou serviço, cartas circulares, correspondências para orçamentos e assim por diante).

ESCOLHA DO HARDWARE

Teoricamente, qualquer microcomputador pode ser usado para processamento de texto, desde que haja um programa adequado para ele (na série de artigos iniciada na página 576, apresentamos um programa completo de processamento de texto para a maioria das linhas cobertas por *INPUT*). É óbvio, porém, que não tem sentido fazer processamento de texto se não se conta com uma impressora. Além disso, convém lembrar que, na prática, existem algumas limitações impostas pelo hardware.

Para começar, há a questão do teclado. A maioria dos micros compatíveis com o ZX-81, por exemplo, possui um teclado inadequado para datilografia rápida de textos, não dispondo de letras minúsculas, nem de sinais de acentuação. Portanto, ao processar textos com essas máquinas, não se pode esperar um resultado de bom nível.

A linha Spectrum original (TK-90X, no Brasil) também tem um teclado sofrível, embora ofereça a alternativa de usar letras minúsculas e algum tipo de acentuação. O modelo TK-95 e os teclados semiprofissionais acessórios, que podem ser comprados para o Spectrum, melhoram bastante a situação.

Os computadores que possuem teclado semelhante ao de uma máquina de escrever — como o Apple, o TK-2000, o TRS-80, o TRS-Color II e o MSX — são mais adequados quanto a esse aspecto. Há, porém, uma grande variação de qualidade entre os diferentes modelos. Deve-se levar em conta, por exemplo, que só os micros da linha MSX incluem a acentuação original da língua portuguesa, por terem adotado o padrão BRASCII (ASCII brasileiro).

Outro elemento importante na escolha do hardware para processamento de texto é o vídeo. Nos computadores domésticos mais comuns, este em geral se limita a 32 ou 40 colunas (excepcionalmente 64, como nos micros TRS-80). Esse formato possibilita, entre outras coisas, o uso de televisores como saída de vídeo. Mas é restritivo para um trabalho mais intenso com processadores de texto, principalmente os do tipo WYSWYG, que imprimem uma folha de sessenta a setenta toques.

Em alguns micros, como o MSX e o Apple, é possível ampliar o formato de vídeo com software ou hardware de expansão. Nesse caso, porém, não se recomenda utilizar aparelhos de TV, pois a imagem perde a nitidez. Para quem pretende trabalhar por muito tempo com um processador, a nitidez de texto e a estabilidade no vídeo são fundamentais — e essas qualidades só são obtidas com monitores profissionais (veja o artigo da página 851).

MEMÓRIA

O tamanho da memória disponível também deve ser considerado na seleção de um bom hardware para processamento de texto. A memória RAM não chega a ser um impedimento, dependendo do software escolhido. Alguns processadores só operam com o texto inteiramente contido na memória. Se ele for demasiado grande, é preciso dividi-lo em seções menores, que são gravadas em fita ou vídeo, o que costuma ser bastante cansativo. Uma maior capacidade da RAM ajuda muito, portanto.

Os processadores mais avançados trabalham à base de *paginação*, ou *memória virtual* — isto é, gravam e lêem essas seções do texto em disco, periodicamente. São, contudo, bem mais exigentes em termos de hardware.

A memória auxiliar é outro item importante. Para quem utiliza pouco o processador e lida apenas com textos pequenos (cartas, digamos), um gravador cassete é perfeitamente aceitável para armazenamento de textos. Para uso mais intenso, entretanto, é praticamente indispensável o acionador de disquetes. Além de maior capacidade e velocidade, ele assegura uma taxa de erros baixa.

O emprego de memória virtual e encadeamento de arquivos (seções separadas de um texto, que podem ser reunidas em qualquer ordem) é necessário quando o texto é limitado apenas pela capacidade da memória auxiliar. Porém, quando os textos são muito grandes, essa solução dificulta o trabalho e impede a utilização de funções globais de edição (busca, substituição etc.)

A IMPRESSORA

Em artigos anteriores (páginas 521 e 648), examinamos os diversos tipos de impressora, suas características e as vantagens que cada modelo oferece em relação ao processamento de texto. Por isso, retomaremos aqui apenas os aspectos mais importantes.

A impressora desempenha um papel fundamental na qualidade do texto impresso (o produto final). Para trabalhos que não requerem uma apresentação perfeita, ou para a obtenção de cópias de "rascunho", uma impressora matricial é perfeitamente adequada. Algumas delas podem produzir impressões em *qualidade carta* (dupla densidade de impressão), com uma aparência bastante semelhante à de um texto datilografado. Se queremos conseguir o máximo de qualidade, porém, devemos usar uma impressora do tipo margarida.

Seja qual for o seu objetivo, não se esqueça de verificar se a impressora é compatível com o software de processamento, sobretudo quanto à capacidade para realce de texto, mudança de fontes, acentuação etc.

QUEM PRECISA DE UM PROCESSADOR?

O processamento de textos por computador oferece enormes vantagens para escritores, jornalistas, professores, cientistas ou quaisquer outros profissionais cujas funções incluam a elaboração de documentos, relatórios e cartas. Se você está em uma dessas categorias e ainda não tem um processador, francamente, não sabemos como pôde passar sem ele, até agora!

Se você é estudante, provavelmente seu grau de utilização não é tão intenso, mas o computador ajuda muito na execução de trabalhos escolares.

De qualquer modo, mesmo que você não se enquadre em nenhum dos casos mencionados, um computador é bem vantajoso em relação a uma máquina de escrever, embora o preço não seja muito diferente (se a máquina for elétrica).

DESENHO ARQUITETÔNICO (2)

Desenhe os móveis que irão compor o ambiente a ser planejado. Depois que todas as peças se encaixarem em seus respectivos lugares, restará salvar o projeto em fita ou disco.

Este é o segundo e último artigo sobre o projeto arquitetônico computadorizado. Aqui estão os complementos das listagens para cada um dos micros, acrescidas de instruções detalhadas referentes às três primeiras opções do menu. As outras quatro serão idênticas para as três versões.

S

As opções são as seguintes:
- opção 1, desenha o ambiente;
- opção 2, posiciona a mobília;
- opção 3, cria mobília própria;
- opção 4, grava o projeto;
- opção 5, carrega;
- opção 6, imprime;
- opção 7, sai do programa.

A primeira informação que devemos fornecer ao programa é a maior medida (em metros) do ambiente considerado. Digite os dados e tecle **<ENTER>**. Depois disso, aparecerá um segundo menu contendo as seguintes opções:
<ESPAÇO> desenha uma linha;
<J> desenha uma janela;
<P> desenha uma porta;
<B> movimenta sem desenhar;
<S> volta ao menu principal.

Ao selecionarmos qualquer opção, à exceção da **<S>**, seremos perguntados sobre a direção e o comprimento da reta a ser feita. As questões serão repetidas, de forma a permitir que tracemos linhas na diagonal, por exemplo, dizendo o quanto para cima ou para a direita queremos a reta. Se desejarmos traçá-la simplesmente na horizontal ou na vertical, bastará que forneçamos valores nulos. Repetiremos o processo até concluir o desenho do ambiente e, em seguida, teclaremos **<S>** para retornar ao menu principal. Com um pouco de prática, desenvolveremos nossos projetos em questão de minutos.

Na opção 2, usaremos as teclas **6** e **7** para selecionar o móvel a ser posicionado. Teclando **<S>**, a peça aparecerá no centro da tela. Poderemos movimentá-la para cima, para baixo, para a direita e para a esquerda acionando as teclas **5**, **6**, **7** e **8**. Para girar o móvel no sentido horário, pressionaremos a tecla **H**; no sentido inverso, **A**.

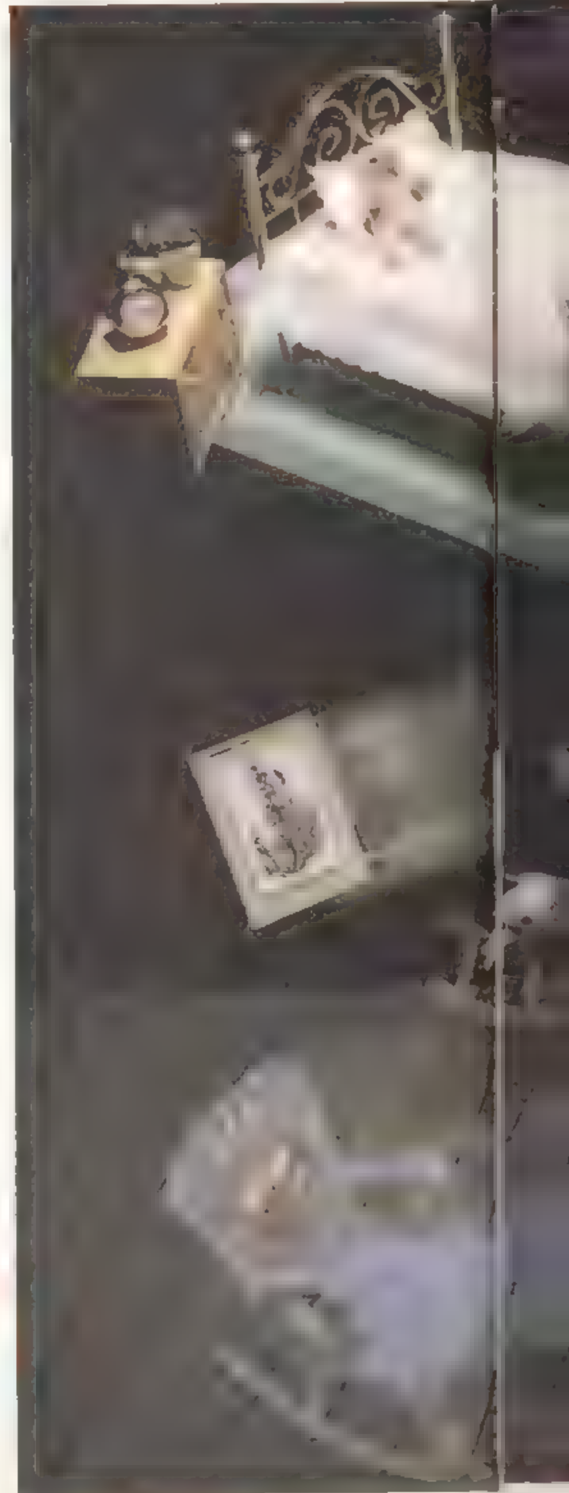
Na criação da mobília, na opção 3, utilizaremos o mesmo processo da opção 1 (desenho do ambiente), com a diferença de que agora o objeto não aparecerá na tela até que sejam fornecidas todas as suas dimensões. Indagados sobre o número de lados da peça (1 a 15) e sobre seu código, daremos um nome de dois dígitos à mobília, de modo a podermos identificá-la mais tarde.

A opção 4 grava tanto a tela como os dados correspondentes às peças da mobília. Usaremos depois a opção 5 para carregar na memória um projeto gravado anteriormente. A opção 6 — de impressão — funcionará somente com a impressora conectada.

```
7010 INPUT "DIGITE DIRECAO  (C.
B.E.D) ?";D$
7020 INPUT "DIGITE DISTANCIA ?"
;D: RETURN
7040 LET Z=CODE INKEY$: FOR K=1
 TO LEN K$: IF CODE (K$(K))=Z T
HEN  LET K=LEN K$: GOTO 7060
7050 NEXT K: GOTO 7040
7060 NEXT K: RETURN
8000 DIM S(10): LET NF=0: LET F
$="000": LET MAX=5: LET R=0: LE
T SC=175/MAX: GOSUB 6060
8010 DIM O(10,30): DIM O$(10,2)
8020 FOR N=1 TO 5: FOR K=1 TO 8
: READ O(N,K): NEXT K  NEXT N
8030 FOR N=1 TO 10: READ O$(N):
 NEXT N
8032 FOR N=USR "A" TO USR "A"+7
: READ A: POKE N,A: NEXT N
8035 FOR N=1 TO 10: READ S(N)
NEXT N
8040 RETURN
8090 DATA .5,0,0,-.5, .5,0,0,.5
8100 DATA .75,0,0,-.5,  75,0,0.
 5
8110 DATA .5,0,0,-.75,-.5,0,0,.
75
8120 DATA .5,0,0,-.5,-.5,0,0,.5
8130 DATA .5,0,0,-.5,-.5,0,0,.5
8150 DATA "AR","FO","LL","PI","
GF","XX","XX","XX","XX","XX"
8160 DATA 16,32,64,255,64,32,16
,0
8170 DATA 4,4,4,4,4,0,0,0,0,0
```

T

Ao rodarmos o programa, veremos um menu com as seguintes opções:
- opção 1, desenha o ambiente;
- opção 2, posiciona a mobília;
- opção 3, cria novas peças;
- opção 4, grava o projeto;



- COMPLETE O DESENHO DO AMBIENTE
- CRIE NOVAS PEÇAS PARA SUA MOBÍLIA
- GRAVE O PROJETO
- CARREGUE
- IMPRIMA
- GIRE SEUS MÓVEIS
- APAGUE E REDESENHE
- PROGRAMA TERMINADO



- opção 5, carrega;
- opção 6, imprime;
- opção 7, sai do programa.

Se escolhermos a opção 1, seremos perguntados sobre a maior medida do ambiente. Com essa informação, o micro colocará o desenho em escala, de modo que ele caiba na tela.

Tecle a barra de espaço para iniciar o desenho do ambiente. Há outras opções dentro da opção 1, a saber:
<ESPAÇO> para desenhar;
<B> para mover sem desenhar;
<W> para desenhar janelas;
<O> para desenhar portas;
<C> para limpar a tela;
<F> para voltar ao menu inicial.

Ao escolher uma das opções (à exceção da **<C>** e da **<F>**), o usuário será indagado sobre a direção e o comprimento da reta desejada. Essas perguntas serão feitas várias vezes, a fim de oferecer dados ao computador.

Suponhamos que queremos traçar uma reta inclinada, por exemplo. Basicamente, teremos que fornecer ao micro duas informações, ou seja, o quanto ela deverá se deslocar para cima ou para baixo e o quanto deverá ir para a direita ou a esquerda. A repetição das questões nos permite, portanto, montar coordenadas bidirecionais para a nossa reta e traçá-la na diagonal.

Para obter retas horizontais ou verticais (unidirecionais), bastará teclar **<ENTER>**, quando a pergunta for repetida. Tecle **<C>** para apagar a tela e **<F>** para voltar ao menu principal.

A opção 2 mantém na tela o desenho do ambiente executado na primeira opção e acrescenta um cursor em forma de cruz que será movimentado ponto a ponto através das teclas das setas. Para movê-lo mais rapidamente, tecle **<SHIFT>** + seta, até alcançar a posição desejada — aquela em que o móvel será colocado. Em seguida, acione a tecla **P**. Feito isso, escolha a peça a ser posicionada. As peças são numeradas de 0 a 9; caberá ao usuário criar as cinco últimas, já que as iniciais são automaticamente definidas (mas podem ser alteradas) no programa: 0-pia, 1-fogão, 2-mesa, 3-geladeira e 4-armário. Para apagar um móvel, posicione o cursor no vértice que o gerou, tecle **D** e selecione o número desse móvel. Deveremos nos certificar de que o cursor esteja exatamente no vértice que o gerou; caso contrário, a peça não será totalmente eliminada

Se quisermos girar um móvel para adequá-lo a um canto qualquer do ambiente, basta acionar a tecla **R** seguido do ângulo de giro do móvel — expresso em graus. Todos os objetos serão desenhados com essa angulação até que se tecle novamente **R**; só que desta vez seguido de 0. A opção **< F >** possibilita retornar ao meu principal.

A opção 3 permite definir novas peças ou modificar as cinco já existentes. As medidas, fornecidas em centímetros, não deverão ultrapassar os 200 cm. O primeiro passo consiste em escolher um





Com o auxílio do computador, pode-se projetar o arranjo de uma cozinha, como mostra a tela à esquerda, em alguns minutos. O menu do programa, exibido na tela abaixo, oferece ao usuário sete opções.

número de 0 a 9 para armazenagem da peça que, em seguida, será desenhada pelo mesmo processo utilizado na opção 1 (desenho do ambiente). Tecle <C> para recomeçar o desenho e <F> para retornar ao menu inicial.

A opção 4 armazena nosso projeto (que deve receber um nome aqui) em fita ou disco. A opção 5 carrega na memória do micro um projeto previamente gravado em fita ou disco.

A opção de impressão está incompleta, pois necessita de uma rotina que despeje o conteúdo da tela na impressora. Retomaremos este assunto em um próximo artigo, que nos habilitará a elaborar tal rotina.

```
1210 OX=X:OY=Y
1220 IF I$=CHR$(8) AND X>0 THEN
 X=X-1
1230 IF I$=CHR$(9) AND X<185 TH
EN X=X+1
1240 IF I$=CHR$(94) AND Y>0 THE
N Y=Y-1
1250 IF I$=CHR$(10) AND Y<185 T
HEN Y=Y+1
1260 IF I$=CHR$(21) AND X>7 THE
N X=X-8
1270 IF I$=CHR$(93) AND X<177 T
HEN X=X+8
1280 IF I$=CHR$(95) AND Y>7 THE
N Y=Y-8
1290 IF I$=CHR$(91) AND Y<177 T
HEN Y=Y+8
1300 PUT(OX-3,OY-3)-(OX+3,OY+3)
,S,PSET
1310 IF I$="P" THEN 1360
1320 IF I$="D" THEN 1430
1330 IF I$="R" THEN 1500
1340 IF I$="F" THEN FOR K=1 TO
4 PCOPY K TO K+4:NEXT:GOTO 90
1350 GOTO 1160
1360 SOUND 190,1
1370 COLOR 0:GOSUB 340
1380 I$=INKEY$:IF I$<"0" OR I$>
"9" THEN 1380
1390 COLOR 1,GOSUB 340
1400 N=VAL(I$)
1410 COLOR 0,1:A$=O$(N):GOSUB 1
540
1420 GOTO 1160
1430 SOUND 200,1
1440 COLOR 0:GOSUB 340
1450 I$=INKEY$:IF I$<"0" OR I$>
"9" THEN 1450
1460 COLOR 1:GOSUB 340
1470 N=VAL(I$)
1480 COLOR 1,1:A$=O$(N):GOSUB 1
540
1490 GOTO 1160
1500 CLS:PRINT"ANGULO DE ROTACA
O (0-360)":INPUT RT
1510 IF RT<0 OR RT>360 THEN 150
0
1520 RT=(RT/180)*3.141
1530 SCREEN 1,0:GOTO 1160
1540 IF A$="" THEN RETURN
1550 P=1:X1=X:Y1=Y
1560 B$=MID$(A$,P,1)
1570 D$=MID$(A$,P+1,1)
1580 D=VAL(MID$(A$,P+2,INSTR(P,
A$,";")-(P+2)))
1590 P=INSTR(P,A$,";")+1
1600 D=D/100:D2=FNA(D):OX=X:OY=
Y
1610 IF D$="D" THEN XA=D2*COS(R
T):YA=-D2*SIN(RT)
1620 IF D$="E" THEN XA=-D2*COS(
RT):YA=D2*SIN(RT)
1630 IF D$="B" THEN XA=D2*SIN(R
T):YA=D2*COS(RT)
1640 IF D$="C" THEN XA=-D2*SIN(
RT):YA=-D2*COS(RT)
1650 XA=INT(XA+.5):YA=INT(YA+.5
)
1660 X=X+XA:Y=Y+YA
1670 IF B$="D" THEN LINE(OX,OY)
-(X,Y),PSET
1680 IF P<LEN(A$) THEN 1560
1690 X=X1:Y=Y1:RETURN
1700 CLS:LINE INPUT"NOME DO ARQ
UIVO:";F$
1710 SG=PEEK(188)*256
1720 CLS:PRINT"SALVANDO:";F$
1730 CSAVEM F$,SG,SG+6143,35252
1740 GOTO 90
1750 CLS LINE INPUT "NOME DO AR
QUIVO:";F$
1760 PRINT"ACIONE O GRAVADOR E
ESPERE"
1770 CLOADM F$
1780 GOTO 90
1790 REM *********************
1791 REM *                   *
1792 REM *    ROTINA PARA    *
1793 REM *  DESPEJAR A TELA  *
1794 REM *   NA IMPRESSORA   *
1795 REM *                   *
1796 REM *********************
1797 GOTO 90
```



Ao rodarmos o programa, veremos na tela um menu com sete opções:

- opção 1, desenha o ambiente;
- opção 2, posiciona a mobília;
- opção 3, cria novas peças;
- opção 4, grava o projeto;
- opção 5, carrega;
- opção 6, imprime;
- opção 7, sai do programa.

Devemos primeiro escolher entre as opções 1, 3 ou 5. Ao decidirmos pela 1, seremos perguntados sobre a maior medida (em metros) do ambiente, informação que permitirá ao micro colocar o desenho em uma escala que o faça caber na tela. Tecle a barra de espaço para iniciar o desenho do ambiente.

Há outras opções dentro da opção 1:
<**ESPAÇO**> para desenhar,
<**B**> para mover sem desenhar;
<**W**> para desenhar janelas;
<**O**> para desenhar portas;
<**C**> para limpar a tela;
<**F**> para voltar ao menu inicial.

Ao escolher uma das opções (à exceção da <C> e da <F>), o usuário será perguntado sobre a direção e o comprimento da reta desejada. Essas perguntas serão feitas várias vezes a fim de oferecer dados ao computador.

Suponhamos que queremos traçar uma reta inclinada, por exemplo. Basicamente, teremos que fornecer ao micro duas informações, ou seja, o quanto ela deverá se deslocar para cima ou para baixo e o quanto deverá ir para a direita ou a esquerda. A repetição das questões permite, portanto, montar coordenadas bidirecionais para nossa reta e traçá-la na diagonal.

Para desenhar retas horizontais ou verticais (unidirecionais), basta teclar <**ENTER**> quando a pergunta for repetida. Tecle <C> para apagar a tela e <F> para voltar ao menu principal.

A opção 2 só poderá ser iniciada depois da opção 1. Ela mantém na tela o desenho do ambiente executado na primeira opção e acrescenta um cursor em forma de seta que será movimentado ponto a ponto através das teclas das setas. Para movê-lo mais rapidamente, usamos as teclas **I**-cima, **M**-baixo, **K**-direita e **J**-esquerda, até que o cursor alcance a posição desejada — aquela em que a peça será colocada. Então acionamos a tecla **P**. Feito isso, determinamos a peça a ser posicionada. As peças são numeradas de 0 a 9; caberá ao usuário criar as cinco últimas peças, já que as iniciais são automaticamente definidas (mas podem ser alteradas) no programa: 0-pia, 1-fogão, 2-mesa, 3-geladeira e 4-armário. Para apagar um móvel, posicione o cursor no vértice que o gerou, tecle **D** e selecione o número correspondente a esse móvel. Deveremos nos certificar de que o cursor esteja exatamente no vértice que o gerou; caso contrário, a peça não será totalmente eliminada.

Se quisermos girar um móvel para fazê-lo caber em algum canto do ambiente, basta acionar **R** seguido do ângulo de giro do móvel, expresso em graus. Todos os objetos serão desenhados com essa angulação, até que se tecle novamente **R**, seguido de 0. A opção <**F**> permite retornar ao menu principal.

A opção 3 possibilita definir novas peças ou modificar as cinco já existentes. As medidas, fornecidas em centímetros, não deverão ultrapassar os 200 cm. O primeiro passo consiste em escolher um número de 0 a 9 para armazenagem da peça que, em seguida, será desenhada pelo mesmo processo utilizado na opção 1 (desenho do ambiente). Tecle <**C**> para recomeçar o desenho, e <**F**> para retornar ao menu inicial.

A opção 4 armazena nosso projeto (que aqui deve receber um nome) em fita. A opção 5 carrega na memória do micro um projeto gravado em fita. Estas duas opções podem ser alteradas para trabalhar com disco, bastando fazer as seguintes modificações: Troque o "**CAS:**" que vem logo após os comandos **BSAVE** e **BLOAD** por "**A:**" (linhas 1720 e 1770), respectivamente.

A opção de impressão está incompleta, pois necessita de uma rotina que despeje o conteúdo da tela na impressora. Retomaremos esse assunto em um próximo artigo, que nos habilitará a elaborar tal rotina.

```
1210 OX=X:OY=Y
1220 IF I$=CHR$(29) AND X>0 THE
N X=X-1
1230 IF I$=CHR$(28)AND X<185THE
N X=X+1
1240 IF I$=CHR$(30) AND Y>0 THE
N Y=Y-1
1250 IF I$=CHR$(31)AND Y<185THE
N Y=Y+1
1260 IF I$=CHR$(74) AND X>7 THE
N X=X-8
1270 IF I$=CHR$(75)AND X<177THE
N X=X+8
1280 IF I$=CHR$(73) AND Y>7 THE
N Y=Y 8
1290 IF I$=CHR$(77)AND Y<177THE
N Y=Y+8
1310 IF I$="P" THEN CL=1·GOTO 1
360
1320 IF I$="D" THEN CL=15·GOTO1
430
1330 IF I$="R" THEN A=USR(0) GO
TO 1500
1340 IF I$="F" THEN A=USR(0)·GO
TO 90
1350 GOTO 1160
1360 BEEP
1370 GOSUB 340
1380 I$=INKEY$·IF I$<"0" OR I$>
"9" THEN 1380
1390 COLOR 0·GOSUB 340
1400 COLOR 1:N=VAL(I$)
1410 A$=O$(N).GOSUB 1540
1420 GOTO 1160
1430 BEEP
1440 GOSUB 340
1450 I$=INKEY$·IF I$<"0" OR I$>
"9" THEN 1450
1460 COLOR 0·GOSUB 340
1470 N=VAL(I$)
1480 A$=O$(N) GOSUB 1540
1490 GOTO 1160
1500 SCREEN0·PRINT"ANGULO DE RO
TAÇAO (0-360)";:INPUT RT
1510 IF RT<0 OR RT>360 THEN 150
0
1520 RT=(RT/180)*3.141:SCREEN 2
1530 A=USR2(0):A=USR1(0):GOTO 1
160
1540 IF A$="" THEN RETURN
1550 P=1:X1=X:Y1=Y
1560 B$=MID$(A$,P,1)
1570 D$=MID$(A$,P+1,1)
1580 D=VAL(MID$(A$,P+2,INSTR(P,
A$,";")-(P+2)))
1590 P=INSTR(P,A$,";")+1
1600 D=D/100:D2=FNA(D):OX=X:OY=
Y
1610 IF D$="D" THEN XA=D2*COS(R
T):YA=-D2*SIN(RT)
1620 IF D$="E" THEN XA=-D2*COS(
RT):YA=D2*SIN(RT)
1630 IF D$="B" THEN XA=D2*SIN(R
T):YA=D2*COS(RT)
1640 IF D$="C" THEN XA=-D2*SIN(
RT):YA=-D2*COS(RT)
1650 XA=INT(XA+.5):YA=INT(YA+.5
)
1660 X=X+XA:Y=Y+YA
1670 IFB$="D" THENLINE(OX,OY)-(
X,Y),CL
1675 COLOR 1
1680 IF P<LEN(A$) THEN 1560
1690 X=X1:Y=Y1·RETURN
1700 LOCATE 5,18·LINE INPUT"NOM
E DO PROJETO:";F$
1710 LOCATE 5,20:PRINT"GRAVANDO
 "+F$+"..."
1720 BSAVE "CAS:"+F$,&HD100,&HE
900
1730 GOTO 90
1750 LOCATE 5,20:LINE INPUT"NOM
E DO PROJETO:";F$
1760 LOCATE 5,22:PRINT"CARREGAN
DO "+F$+"..."
1770 BLOAD "CAS:"+F$
1780 F1=1:GOTO 90
1790 REM **********************
1791 REM *                    *
1792 REM *   rotina para      *
1793 REM * despejar a tela    *
1794 REM *  na impressora     *
1795 REM *                    *
1796 REM **********************
1797 GOTO 90
2000 CLEAR 200,&HA000
2010 DEFUSR=&HA000
2015 DEFUSR1=&HA00D
2020 DEFUSR2=&HA01A
2025 DEFUSR3=&HA026
2030 DEFUSR4=&HA033
2040 FOR R=0 TO 63:READ A
2050 POKE(&HA000+R),A:NEXT
2060 DATA &H21,00,00,&H11,00,&H
D1,01,00,24,&HCD,&H59,00,&HC9
2070 DATA &H21,00,&HD1,&H11,00,
00,01,00,24,&HCD,&H5C,00,&HC9
2080 DATA &H21,00,&H20,62,00,01
,00,24,&HCD,&H56,00,&HC9
2090 DATA &H21,00,00,&H11,&HFF,
&HB8,01,00,24,&HCD,&H59,00,&HC9
2100 DATA &H21,&HFF,&HB8,&H11,0
0,00,01,00,24,&HCD,&H5C,00,&HC9
2110 POKE(&HA01E),31
2120 SP$="":FORI=1TO8:READA$
2130 SP$=SP$+CHR$(VAL("&H"+A$))
2140 NEXT·SPRITE$(0)=SP$
2150 GOTO 20
2160 DATA E0,C0,A0,90,08,00,00,
00
```

DESENHOS EM PERSPECTIVA

- PERSPECTIVA
- PONTO DE FUGA E PONTO DE VISTA
- DIMINUIÇÃO DE TAMANHO
- TÉCNICA DE SOMBREAMENTO



Traga seu desenho para a magia do mundo tridimensional. Isso é possível graças aos recursos de seu micro aliados aos princípios da perspectiva e das técnicas de sombreamento.

A maioria das pessoas sabe da importância do uso da perspectiva em qualquer figura, e, por isso, costuma incluí-la até nos mais simples esboços. Surpreendentemente, a idéia de perspectiva não é tão clara como parece; assim, os artistas mais antigos não tinham a menor idéia sobre o que fazer para dar profundidade às suas telas. Apenas a partir do Renascimento, quando os pintores começaram a dar um tom realista às representações, é que os princípios de perspectiva foram formulados. Essas regras são as mesmas usadas pelos artistas contemporâneos e aplicadas aos desenhos feitos no computador.

Para o desenvolvimento dessa técnica, foram necessários vários séculos, o que não significa que suas noções básicas sejam de difícil compreensão. Na realidade, a idéia fundamental é a seguinte: todas as linhas horizontais permanecem nessa posição, enquanto as verticais se inclinam "para dentro" da figura, convergindo para o denominado *ponto de fuga*. (Esse assunto já foi abordado no artigo *Programação de gráficos em 3-D*, página 693.)

Apenas as linhas paralelas, perpendiculares à figura, se dirigem para o ponto de fuga principal. As demais afluem para seus próprios pontos de fuga, situados em uma mesma linha, no horizonte imaginário do desenho.

Esses princípios, quando aplicados a figuras reais, podem se tornar um pouco mais complicados; em desenhos simples, entretanto, proporcionam a sensação de profundidade exigida. O leitor já deve ter visto, na televisão ou no cinema, astronaves voando para o espaço a partir do imenso hangar da nave-mãe. Nesse caso, a sensação de profundidade é dada pelo chão e pelo teto quadriculados do hangar, que parecem se projetar em direção ao espaço.

FEIXES DE RETA EM PERSPECTIVA

Os feixes de retas perpendiculares são muito usados para se obter a visão de

um objeto em perspectiva. O primeiro programa, mostrado a seguir, desenha duas superfícies quadriculadas — o teto e o chão do hangar — tendo o usuário do micro como observador, olhando para o infinito.

```
 10 BORDER 0· PAPER 0: INK 7
 20 CLS : INPUT "PROXIMIDADE D
 O CHAO ";V
 40 FOR K=-126 TO 127 STEP 6
 50 LET Y=174: LET X=K+V*K: IF
 ABS (X)>127+(X<1) THEN  LET X
 =SGN (X)*127-(X<1): LET Y=100
 +(X-K)*74/(V*K)
 60 PLOT 127-K,100· DRAW 127-X
 -PEEK 23677,Y-PEEK 23678
 70 PLOT 127-K,74: DRAW 127-X-
 PEEK 23677,174-Y-PEEK 23678
 80 NEXT K
 90 LET F=V^(1/6): LET Y=F
100 LET Y=Y*F: IF Y>77 THEN
GOTO 140
110 PLOT 0,97+Y: DRAW 255,0
120 PLOT 0,77-Y: DRAW 255,0
130 GOTO 100
140 IF INKEYS="" THEN  GOTO
 140
150 GOTO 20
```

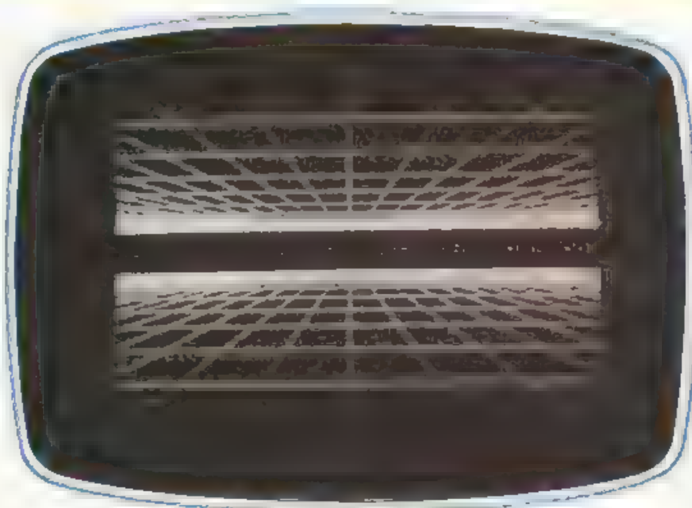
```
10 PMODE 4,1
20 CLS:INPUT"PROXIMIDADE DO CHA
O ",V
30 PCLS:SCREEN 1,1
40 FOR K=-126 TO 127 STEP 6
50 Y=191·X=K+V*K·IF ABS(X)>127-
(X<1) THEN X=SGN(X)*127+(X<1):Y
=111+(X K)*80/(V*K)
60 LINE(127 K,111) (127-X,Y),PS
ET
70 LINE(127 K,80)-(127-X,191 Y)
,PSET
80 NEXT
90 F=V↑(1/6):Y=F
100 Y=Y*F:IF Y>83 THEN 140
110 LINE(0,108+Y)-(255,108+Y),P
SET
120 LINE(0,84-Y)-(255,84-Y),PSE
T
130 GOTO 100
140 IF INKEYS="" THEN 140 ELSE
20
```



```
10  TEXT
20  HOME : INPUT "PROXIMIDADE D
O CHAO ";V
30  HGR2
40  FOR K =  - 126 TO 127 STEP
6
50 Y = 191:X = K + V * K· IF  A
BS (X) > 127 - (X < 1) THEN X =
 SGN (X) * 127 + (X < 1):Y = 1
11 + (X - K) * 80 / (V * K)
60  HPLOT 127 - K,111 TO 127 -
X,Y
70  HPLOT 127 - K,80 TO 127 - X
,191 - Y
80  NEXT
90 F = V ^ (1 / 6):Y = F
100 Y = Y * F: IF Y > 83 THEN 1
40
110  HPLOT 0,108 + Y TO 255,108
 + Y
120  HPLOT 0,84 - Y TO 255,84 -
 Y
130  GOTO 100
140  GET A$: GOTO 10
```

```
10 SCREEN 0
20 CLS:INPUT"PROXIMIDADE DO CHA
O ";V
30 SCREEN 2
40 FOR K=-126 TO 127 STEP 6
50 Y=191:X=K+V*K:IF ABS(X)>127-
(X<1) THEN X=SGN(X)*127+(X<1):Y
=111+(X-K)*80/(V*K)
60 LINE(127-K,111)-(127-X,Y)
70 LINE(127-K,80)-(127-X,191-Y)
80 NEXT
90 F=V^(1/6):Y=F
100 Y=Y*F:IF Y>83 THEN 140
110 LINE(0,108+Y)-(255,108+Y)
120 LINE(0,84-Y)-(255,84-Y)
130 GOTO 100
140 IF INKEY$="" THEN 140 ELSE
20
```

Alterando o valor inicial, obtêm-se efeitos diferentes, como a sensação de estar flutuando entre as duas superfícies. Atribuindo um valor baixo para **V** (2, por exemplo), as linhas inferiores se inclinarão para os pontos de fuga situados em um horizonte acima da metade do vídeo. Inversamente, as linhas superiores convergirão para um horizonte abaixo da metade do vídeo. Se, por outro lado, digitarmos o valor 10, o cenário apresentará modificações profundas, com o observador em uma posição bem mais próxima do chão.



Um cenário quadriculado em perspectiva é um bom recurso para conferir profundidade a seu desenho.

Inicialmente, o programa desenha as retas frontais voltadas para o horizonte, entre as linhas 40 e 80, e as retas que vão de lado a lado da tela, entre as linhas 90 e 130. A variável **K** dá a coordenada X do início de cada linha, que é, então, multiplicada por **V** para chegar à coordenada X da sua extremidade final. A condição **IF...THEN**, na linha 50, apenas evita que o desenho ultrapasse os limites da tela. Em alguns computadores, isto é dispensável, apesar de acelerar a execução do programa.

As linhas restantes serão traçadas pela próxima parte do programa. A distância entre elas torna-se cada vez menor à medida que "se afastam" da tela, um efeito controlado por sucessivas multiplicações da coordenada Y por **V↑1/6**. O valor 1/6 foi escolhido para dar um resultado mais realista; isto, porém, não impede que o usuário experimente outros valores.

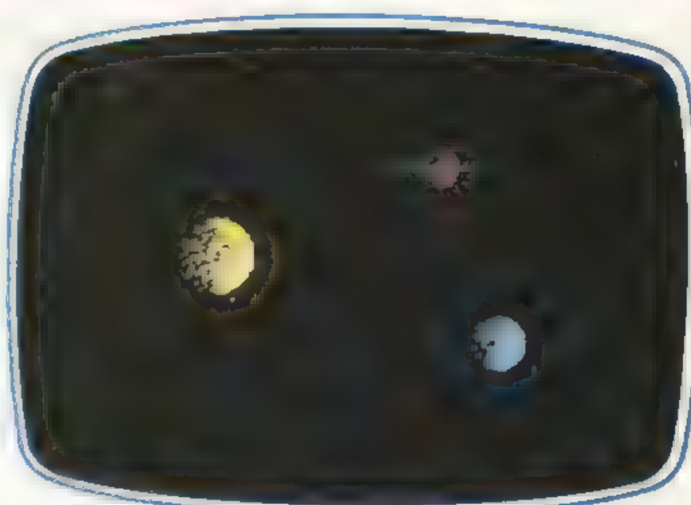
A DIMINUIÇÃO DE TAMANHO

Os princípios da perspectiva aplicam-se aos objetos bem como aos feixes de retas. Vistos a distância, todos os elementos de um desenho parecem menores e mais próximos uns dos outros. Isso se verifica, por exemplo, quando observamos os topos e as bases de uma série de árvores iguais enfileiradas colocadas sobre duas retas que convergem para um mesmo ponto de fuga.

Obtém-se esse efeito porque o cére-



À medida que nos afastamos dos objetos, eles parecem menores e mais próximos entre si, como mostra a figura acima.



São necessários princípios e técnicas especiais para se obter um bom sombreamento em esferas.

bro humano relaciona o tamanho de um objeto conhecido com a distância em que ele se encontra. Ao vermos dois objetos idênticos na forma, mas um com a metade do tamanho do outro, deduziremos que o menor está duas vezes mais longe que o primeiro. Portanto, ao desenharmos alguns objetos em tamanho decrescente no vídeo de um micro, teremos a impressão de que eles estarão "entrando na tela".

Analisando a distância entre os objetos de uma figura, concluiremos que ela se comporta do mesmo modo. Os objetos parecem mais próximos entre si à medida que se afastam do observador.

Existe uma relação matemática entre a distância e o tamanho, isto é, o tamanho apresenta uma variação *inversamente* proporcional à distância. Assim, se aumentarmos a distância entre o objeto e o observador, o tamanho do objeto diminuirá. É por essa razão que no primeiro programa a separação entre as linhas horizontais está elevada a 1/6. O número 6 foi escolhido apenas para se garantir um espaçamento mais condizente com a realidade; se o substituirmos por qualquer outro, veremos que o desenho não perderá sua profundidade.

O próximo programa mostra como desenhar uma vista em perspectiva de uma estrada delimitada por postes:

S

```
10 BORDER 0: PAPER 0: INK 7:
CLS
15 DEF FN Y(X)=((174-VP)/100)
*(X-128)+VP
16 DEF FN B(X)=((20-VP)/100)*
(X-128)+VP
17 DEF FN S(T)=SF/SQR ((RW/2)
^2+(T*PH)^2)
20 PRINT '''
30 INPUT "DISTANCIA ENTRE POS
TES ";P
40 INPUT "LARGURA DA ESTRADA
";RW
50 INPUT "ALTURA DOS POSTES "
;PH
60 INPUT "ALTURA DO OBSERVADO
R ";RH
70 CLS
80 LET SF=1: LET SF=160/FN S(
0)
90 LET VP=160/PH*RH+100: LET
X=228: FOR T=1 TO 15
100 LET X=X-1: IF FN S(T)<FN Y
(X)-FN B(X) THEN  GOTO 100
110 PLOT X,FN B(X): DRAW X-
PEEK 23677,FN Y(X)-PEEK 23678:
LET XJ=(FN Y(X)-FN B(X))/10:
LET YJ=FN Y(X)-XJ
120 PLOT X-XJ,FN Y(X): DRAW X-
XJ-PEEK 23677,YJ-PEEK 23678:
DRAW X+XJ-PEEK 23677,YJ-PEEK
23678: DRAW X+XJ-PEEK 23677,FN
Y(X)-PEEK 23678
130 PLOT 255-X,FN B(X): DRAW
255-X-PEEK 23677,FN Y(X)-PEEK
23678
140 PLOT 255-X-XJ,FN Y(X):
DRAW 255-X-XJ-PEEK 23677,YJ-
PEEK 23678: DRAW 255 X+XJ-PEEK
23677,YJ-PEEK 23678: DRAW 255-
X+XJ-PEEK 23677,FN Y(X)-PEEK
23678
150 NEXT T
160 GOTO 160
```

T

```
10 PMODE 4,1:PCLS:CLS
20 DEFFNYT(X)=((900-VP)/500)*(X
-640)+VP:DEFFNYB(X)=((100-VP)/5
00)*(X-640)+VP:DEFFNS(T)=SF/SQR
((RW/2)↑2+(T*PH)↑2)
30 INPUT"DISTANCIA ENTRE OS POS
TES";P
40 INPUT"LARGURA DA ESTRADA";RW
50 INPUT"ALTURA DOS POSTES";PH
60 INPUT"ALTURA DO OBSERVADOR";
RH
70 SCREEN 1,1
80 SF=1:SF=800/FNS(0)
90 VP=800/PH*RH+100:X=1140:FOR
T=1 TO 15
100 X=X-4:IF FNS(T)<FNYT(X)-FNY
B(X) THEN 100
110 LINE(X/5,191-FNYB(X)/5)-(X/
5,191-FNYT(X)/5),PSET:XJ=(FNYT(
X)-FNYB(X))/10:YJ=FNYT(X)-XJ
120 LINE((X-XJ)/5,191-FNYT(X)/5
)-((X-XJ)/5,191-YJ/5),PSET:LINE
-((X+XJ)/5,191-YJ/5),PSET:LINE-
((X+XJ)/5,191-FNYT(X)/5),PSET
130 LINE(255-X/5,191-FNYB(X)/5)
-(255-X/5,191-FNYT(X)/5),PSET
140 LINE(255-(X+XJ)/5,191-FNYT(
X)/5)-(255-(X+XJ)/5,191-YJ/5),P
SET:LINE -(255-(X-XJ)/5,191-YJ/
5),PSET:LINE-(255-(X-XJ)/5,191-
YJ/5),PSET:LINE-(255-(X-XJ)/5,1
91-FNYT(X)/5),PSET
150 NEXT
160 IF INKEY$="" THEN 160 ELSE
RUN
```



```
10  TEXT : HOME
20  DEF  FN YT(X) = ((900 - VP)
 / 500) * (X - 640) + VP: DEF
FN YB(X) = ((100 - VP) / 500) *
 (X - 640) + VP: DEF  FN S(T) =
 SF /  SQR ((RW / 2) ^ 2 + (T *
 PH) ^ 2)
30  INPUT "DISTANCIA ENTRE OS P
OSTES ";P
40  INPUT "LARGURA DA ESTRADA "
;RW
50  INPUT "ALTURA DOS POSTES ":
PH
60  INPUT "ALTURA DO OBSERVADOR
 ";RH
70  HCOLOR= 3: HGR2
```

```
80 SF = 1:SF = 800 /  FN S(0)
90 VP = 800 / PH * RH + 100:X =
 1140: FOR T = 1 TO 15
100 X = X - 4: IF  FN S(T) <  F
N YT(X) -  FN YB(X) THEN 100
110  HPLOT X / 5,191 -  FN YB(X
) / 5 TO X / 5,191 -  FN YT(X)
/ 5:XJ = ( FN YT(X) -  FN YB(X)
) / 10:YJ =  FN YT(X) - XJ
120  HPLOT (X - XJ) / 5,191 -
FN YT(X) / 5 TO (X - XJ) / 5,19
1 - YJ / 5: HPLOT  TO (X + XJ)
/ 5,191 - YJ / 5: HPLOT  TO (X
+ XJ) / 5,191 -  FN YT(X) / 5
130  HPLOT 255 - X / 5,191 -  F
N YB(X) / 5 TO 255 - X / 5,191
-  FN YT(X) / 5
140  HPLOT 255 - (X + XJ) / 5,1
91 -  FN YT(X) / 5 TO 255 - (X
+ XJ) / 5,191 - YJ / 5: HPLOT
TO 255 - (X - XJ) / 5,191 - YJ
/ 5: HPLOT  TO 255 - (X - XJ) /
 5,191 -  FN YT(X) / 5
150  NEXT
160  GET A$: RUN
```



```
10 SCREEN 0
20 DEFFNYT(X)=((900-VP)/500)*(X
-640)+VP:DEFFNYB(X)=((100-VP)/5
00)*(X-640)+VP:DEFFNS(T)=SF/SQR
((RW/2)^2+(T*PH)^2)
30 INPUT"DISTANCIA ENTRE OS POS
TES ";P
40 INPUT"LARGURA DA ESTRADA ";R
W
50 INPUT"ALTURA DOS POSTES ";PH
60 INPUT"ALTURA DO OBSERVADOR "
;RH
70 SCREEN 2
80 SF=1:SF=800/FNS(0)
90 VP=800/PH*RH+100:X=1140:FOR
T=1 TO 15
100 X=X-4:IF FNS(T)<FNYT(X)-FNY
B(X) THEN 100
110 LINE (X/5,191-FNYB(X)/5)-(X
/5,191-FNYT(X)/5):XJ=(FNYT(X)-F
NYB(X))/10:YJ=FNYT(X)-XJ
120 LINE ((X-XJ)/5,191-FNYT(X)/
5)-((X-XJ)/5,191-YJ/5):LINE-((X
+XJ)/5,191-YJ/5):LINE-((X+XJ)/5
,191-FNYT(X)/5)
130 LINE (255-X/5,191-FNYB(X)/5
)-(255-X/5,191-FNYT(X)/5)
140 LINE (255-(X+XJ)/5,191-FNYT
(X)/5)-(255-(X+XJ)/5,191-YJ/5):
LINE-(255-(X-XJ)/5,191-YJ/5):LI
NE-(255-(X-XJ)/5,191-FNYT(X)/5)
150 NEXT
160 IF INKEY$="" THEN 160 ELSE
RUN
```

Este programa permite ao usuário definir exatamente o cenário da estrada antes de desenhá-lo em perspectiva.

Para facilitar os cálculos, ele oferece três funções: **FNS(T)** calcula a altura de cada poste; **FNYT(X)** e **FNYB(X)** dão as coordenadas Y da extremidade inferior e superior dos postes para qualquer posição X. Podemos imaginar o trabalho dessas duas funções como sendo o de traçar duas retas que afluem para um determinado ponto de fuga, entre as quais serão colocados os postes.

Na parte principal do programa, a linha 80 calcula o fator de escala, **SF**, que constrói um poste tomando por base a altura do anteriormente desenhado. A variável **VP** define o valor da coordenada Y do ponto de fuga, que, por sua vez, depende da altura do observador e da altura dos postes. A coordenada X está sempre no centro da tela.

Quinze postes são desenhados em cada lado da estrada pelo laço entre as linhas 90 e 150. Os da margem direita são os primeiros a serem erguidos.

Inicialmente, a linha 100, começando a partir da extremidade direita da tela, decresce o valor da posição X, até que a altura do poste, calculada por **FNS(T)**, se encaixe exatamente entre as linhas criadas por **FNYT** e **FNYB**. Feito isso, a primeira parte da linha 110 desenha o poste, enquanto a porção restante manipula as variáveis **XJ** e **YJ**, que são fatores de escala para a construção da parte superior (suporte dos fios). A linha 120 é responsável por essa última tarefa.

SOMBREAMENTO

Além da perspectiva, existem outras técnicas capazes de dar tridimensionalidade aos gráficos. A mais comum delas é o sombreamento. Infelizmente, a falta de variação de tons das cores constitui um grande problema quando tentamos aplicar essa técnica aos computadores domésticos. Com isso, torna-se extremamente difícil criar o sombreamento necessário para um desenho em três dimensões.

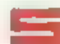
Já os computadores maiores, oferecendo uma rica gama de tonalidades, fazem dessa tarefa algo bem mais fácil.

PLANETAS PONTILHADOS

É possível acrescentar sombras a uma figura qualquer colorindo apenas alguns pontos em sua região mais escura e a maioria deles nas partes mais claras. Portanto, o número de pontos acesos em uma certa área do objeto é que irá determinar a claridade.

Com a variação gradual da densidade nas diferentes regiões, poderemos criar o efeito de sombreamento. O próximo programa utiliza essa técnica para desenhar um grupo de esferas. Elas se parecem tanto com planetas girando no espaço que o programa chega a concluir um fundo estrelado e a desenhar um anel ao redor de algumas delas.





```
10 BRIGHT 0: BORDER 0: PAPER
0: INK 7: CLS
20 FOR I=1 TO 100: PLOT RND*
255,RND*175: NEXT I
25 LET F=0: FOR T=1 TO 3
30 LET CL=RND*5+2
40 LET XC=RND*195+30: LET YC=
RND*115+30
50 LET S=RND*30
60 FOR K=-S TO S
70 IF INT (K)=0 AND F=0 THEN
PLOT  INVERSE 1; OVER 1;XC-L*
2,YC: DRAW  INK CL;L*4,0: LET
F=1
80 LET X=SQR (S*S-K*K)
90 LET X2=2*X
100 FOR L=-X TO X
110 PLOT  INK CL; INVERSE 1;XC
+L,YC+K: IF RND*X2-X<L THEN
PLOT  INK CL;XC+L,YC+K
120 NEXT L
130 NEXT K
140 NEXT T
```



```
150 PAUSE 0
```



```
10 PMODE 3,1:PCLS3:SCREEN 1,0
20 FOR K=1 TO 100:PSET(RND(256)
-1,RND(192)-1,2):NEXT
25 FOR T=1 TO 8
30 XC=RND(195)+30:YC=RND(131)+3
0
40 CL=RND(3):CL=CL-(CL=3)
50 S=RND(25)+5
60 FOR K=-S TO S
70 IF K=1 AND RND(4)=1 THEN CIR
CLE(XC,YC),S*1.5,RND(4),0
80 X=SQR(S*S-K*K)
90 X2=2*X
100 FOR L=-X TO X STEP 2
110 IF RND(X2)-X<L THEN COLOR C
L ELSE COLOR 3
120 PSET(XC+L,YC-K)
130 NEXT L,K
140 NEXT T
150 GOTO 150
```



```
10  HCOLOR= 3: HGR2
20  FOR K = 1 TO 100: HPLOT  IN
T (279 *  RND (1) + 1), INT (19
1 *  RND (1) + 1): NEXT K
25  FOR T = 1 TO 8
30 XC =  INT ( RND (1) * 195 +
30)·YC =  INT ( RND (1) * 131 +
 30)
40 CL =  INT ( RND (0) * 3 + 1)
:CL = CL - (CL = 3)
50 S =  INT ( RND (1) * 25 + 5)

60  FOR K =  - S TO S
70  IF K = 1 AND  INT ( RND (1)
 * 4 + 1) = 1 THEN  GOSUB 160
80 X =  SQR·(S * S - K * K)
90 X2 = 2 * X
100  FOR L =  - X TO X STEP 2
110  IF  INT ( RND (0) * X2 + 1
) - X < L THEN  HCOLOR= CL: GOT
O 120
115  HCOLOR= 3
120  HPLOT XC + L,YC - K
130  NEXT L,K
140  NEXT T
150  GOTO 150
160  FOR A = 0 TO 6.28 STEP  05

170  HPLOT S * 2 *  COS (A) + X
C,S *  5 *  SIN (A) + YC
180  NEXT A
190  RETURN
```



```
10 COLOR 15,1,1·SCREEN 2
20 FOR K=1 TO 100:PSET(INT(RND(
1)*256),INT(RND(1)*192)):NEXT
25 FOR T=1 TO 8
30 XC=INT(RND(-TIME)*195)+30·YC
=INT(RND(-TIME)*131)+30
40 CL=INT(RND(1)*3):CL=CL-(CL=3
)
50 S=INT(RND(1)*25)+5
60 FOR K=-S TO S
70 IF K=1 AND INT(RND(1)*4)=1 T
HEN CIRCLE(XC,YC),S*1.5,9,0,,.2
5
80 X=SQR(S*S-K*K)
90 X2=2*X
100 FOR L=-X TO X STEP 2
110 IF INT(RND(1)*X2) X<L THEN
COLOR CL ELSE COLOR 3
120 PSET(XC-L,YC-K)
130 NEXT L,K
140 NEXT T
150 GOTO 150
```

O cenário de estrelas é feito pela linha 20, que acende aleatoriamente 100 pontos na tela. Oito esferas serão desenhadas (nos micros da linha Spectrum apenas três) pelo laço entre as linhas 25 e 140. A linha 30 escolhe uma posição casual para o centro de cada esfera, e as linhas seguintes determinam cor e tamanho.

O procedimento geral para a criação de cada esfera consiste em preenchê-la, de baixo para cima, com retas horizontais. Esse trabalho é controlado por meio da variável **K**, que tem −S como valor inicial (a coordenada Y da parte mais baixa da esfera) e +S como valor final (a coordenada Y da parte mais alta da esfera).

A linha 80 define a coordenada X do início de cada reta através da equação para construção do círculo. **X2** é o comprimento de cada reta.

As linhas 100 e 130 determinam a cor das linhas. A variavel L pode ser considerada como a luminosidade de cada região; seu valor varia segundo a distância entre o ponto e a borda do círculo. Um número aleatório (dependendo do comprimento da reta) é escolhido pela linha 110; caso seja menor que a claridade da região, define-se o ponto a ser plotado como sendo de cor preta. Se, ao contrário, o número for maior, o ponto será definido como sendo da cor escolhida na linha 40

Por fim, a linha 135 desenha um anel ao redor de dois ou três planetas.

Experimente adaptar estas rotinas para sombrear outros sólidos, como cilindros ou superfícies planas.

FORMATAÇÃO DE TELAS

Aprenda a montar telas para entrada de dados formatados nos computadores Apple e TK-2000. A padronização tornará seu trabalho mais profissional e confiável.

Se você já se deu ao trabalho de escrever um programa, certamente enfrentou dificuldades na padronização da entrada dos dados. Quantas vezes não foi preciso verificar o número máximo de caracteres de uma cadeia *string*, para que não excedesse o tamanho do seu arquivo, ou manipular uma data para dar-lhe um formato adequado?

A rotina que publicamos aqui resolve esses e outros problemas, padronizando o formato de entrada de dados e, ainda, melhorando sua apresentação. Ela fornece formatos definidos para a entrada de dados numéricos, alfanuméricos, data e hora. Além disso, para que você tenha uma preocupação a menos, encarrega-se de verificar a validade das datas e horas.

COMO USAR A ROTINA

A utilização da rotina de formatação é simples. Digite o programa e execute-o, para ter uma amostra do que ela faz. Antes de chamar a rotina, você precisará apenas entrar um parâmetro que definirá o formato e o tipo de dado que a máquina aceitará.

O parâmetro será fornecido pela variável **P$**, já que o BASIC não permite a passagem direta de parâmetros para as sub-rotinas. Assim, se você quiser dar entrada a uma data, a variável **P$** deverá conter a letra C (faça **P$** = "C" e use "calendário" como mnemônico); para a entrada de hora, **P$** conterá o valor H (**P$** = "H"). Como esses dois formatos são fixos, basta uma só letra.

Para a entrada de números e de variáveis alfanuméricas, mais informações serão necessárias. Na entrada de números, a variável **P$** deverá conter a letra D seguida de dois valores, que indicam o número de algarismos (incluindo o ponto decimal) e o número de casas decimais. O primeiro valor deve ter sempre dois dígitos (até 9, coloque 0 antes do algarismo significativo); o segundo, apenas um dígito (use uma vírgula para separá-lo do anterior). Se você quiser, por exemplo, dar entrada a um número de nove dígitos, dois dos quais decimais, use: **P$** = "D09,2". Para a entrada de variáveis alfanuméricas, utiliza-se a letra G (lembre-se de "geral", já que aqui são aceitos números e letras), seguida do número máximo de caracteres a serem digitados. Para a entrada de um nome de até trinta caracteres, use: **P$** = "G,30".

Definido o parâmetro, chama-se a sub-rotina por meio do comando **GOSUB 600**. Se for conveniente, você poderá renumerá-la e colocá-la em qualquer parte do seu programa. Nesse caso, não se esqueça de mudar o número da linha pedida pelo **GOSUB**.

Depois de digitada, a informação é armazenada na variável **W$**. Para usar seu conteúdo, é necessário transferi-lo para outra variável cujo valor não seja alterado pela sub-rotina. Uma linha de programa que utiliza a rotina de entrada de dados fica assim:

```
100 PRINT "VALOR DO SALARIO";:
P$="D08,2":GOSUB 600:SA=VAL(W$)
```

Quanto à localização do formato para a entrada de dados na tela, a rotina também é simples. Seu ponto inicial será aquele em que você tiver deixado o cursor antes de chamar a rotina. Para marcar esse ponto, coloque um ponto e vírgula no fim da mensagem. Se essa posição não lhe convier, posicione o cursor com **HTAB** e **VTAB**.

Como está, a rotina não trabalha bem se o formato ficar dividido em duas linhas, pois só o posicionamento horizontal é monitorado. Mais à frente, veremos como modificar essa situação.

FUNCIONAMENTO

A rotina começa na linha 600, onde a variável **PH** passa a conter o valor da atual posição horizontal do cursor. A posição de memória 36 armazena valores de 0 a 39. Somamos 1 ao valor em questão, tornando-o equivalente ao usado por **HTAB**. A variável **PP** passa então a conter o valor ASCII da letra que caracteriza o parâmetro. Com base nessa variável, o programa será desviado adequadamente na linha 604. As linhas indicadas em 604 atribuem os valores necessários às variáveis que controlam o funcionamento da sub-rotina.

Examinemos essas variáveis. Inicialmente, **P** contém o comprimento do dado a ser digitado. Para data e hora, esse valor é fixo; para outros tipos de dado, é lido de **P$**.

PF indica o ponto em que se poderá deixar a rotina. Assim, para entradas alfanuméricas, seu conteúdo é 1 — ou seja, pelo menos um caractere tem que ser digitado, não se aceitando entradas vazias. Nos demais casos, **PF** é igual a **P**, forçando o uso de todos os campos do formato de entrada. Isso impede que uma data seja digitada pela metade ou que se omita a parte decimal de um número (mesmo que seja 0).

Duas outras variáveis, **P1** e **P2**, marcam a posição onde se usam caracteres fixos no formato. Assim, para datas, o caractere "/" é utilizado duas vezes. A posição é contada a partir de 0.

Finalmente, as variáveis alfanuméricas iniciadas por W servem para montar a linha de formatação, que ficará armazenada em **WD$**.

Depois que o conteúdo dessas variáveis foi definido, a execução converge para a linha 618. Aí terá início a entrada de dados propriamente dita. O formato é colocado na posição lida pela linha 600 e as variáveis **W$**, **WI$** e **PX** são limpas, passando a conter a posição do caractere dentro do *string* que está sendo trabalhado.

A linha 626 indica o caractere correspondente à posição atual do cursor, usando operações lógicas. Para isso, o computador atribui valor 1 às relações verdadeiras, e 0, às falsas. Assim, todas as comparações são condensadas em uma única linha, evitando-se o uso de várias instruções **IF**.

O caractere indicado é colocado em sua posição e, em seguida, o cursor é reposicionado. Espera-se então a entrada de um caractere pelo teclado (linha 628). A rotina só particulariza dois tipos de caractere: o *carriage return* (ou seja, <**CR**>, cujo código é 13) e o *backspace* (ou ←, de código 8). Os demais são tratados em grupo.

Ao se digitar <**CR**>, o caractere será reconhecido pelo seu código, que está armazenado na variável **PA**. Note que ele só terá efeito se a posição do caractere atual (**PX**) for maior ou igual ao

- UMA ROTINA PARA ENTRADA DE DADOS
- DEFINIÇÃO DO PARÂMETRO
- TIPOS DE DADOS
- ARMAZENAGEM DA INFORMAÇÃO
- POSICIONAMENTO DO CURSOR
- COMO FUNCIONA
- APERFEIÇOAMENTOS

ponto de saída definido (**PF**). Assim, se estivermos entrando uma data, não poderemos deixar a rotina antes de digitar o último caractere.

O back-space (linha 632) só terá efeito se estiver além do primeiro caractere (**PX>0**). Nesse caso, a rotina é desviada para a linha 644 e o valor 1 é subtraído de **PH** — ou seja, a tabulação horizontal retrocede uma posição. Se **PX** está apagando o primeiro caractere (**PX=1**), volta-se para a linha 620, e o processo recomeça. Na linha seguinte, **PX** é decrementado de 1, pois estamos retornando a uma posição anterior. Verifica-se então se o cursor não caiu sobre um dos caracteres fixos do formato (cujas posições são representadas por **P1** e **P2**). Em caso afirmativo, recua-se o cursor mais uma posição. Na linha 648, o conteúdo de **W$** (onde a entrada está sendo montada) é atualizado e retoma-se o ciclo.

Não sendo **<CR>** ou ←, o caractere digitado é interpretado como uma entrada de dado. As linhas 634 a 638 checam se ela é válida. Para valores de **PP** iguais a 1, 2 e 6 (3 e 4 não são usados), apenas números são admitidos. Para **PP** igual a 5 (entradas alfanuméricas) admitem-se letras maiúsculas e números. Se o caractere não for válido, a execução é devolvida à linha 628. Caso contrário, a linha 640 se encarrega de colocá-lo na tela e adicioná-lo à variável **W$**, incrementando, em seguida, **PH** e **PX**. Verifica-se então se o cursor foi para uma posição de caractere fixo, que deve ser saltado.

Após todos esses passos, retornamos à linha 624, que encerra a entrada quando o último caractere do dado é digitado, desviando a execução para a linha 650. Esta linha e a 652 voltam a impressão de tela para o modo **NORMAL** e apagam a parte não utilizada do formato. Nesse ponto, a rotina de entrada de dados está terminada. Segue-se a checagem da validade de datas e horas por meio de operações lógicas.

APERFEIÇOAMENTOS

Podemos agora fazer algumas modificações na rotina — por exemplo, transferir a declaração da variável **WW$** da linha 601 para o início do programa, visto que só é necessário defini-la uma vez, e não sempre que a rotina for utilizada. Talvez você não precise usar todos os tipos de entrada. Nesse caso, é possível retirar as partes dispensáveis da rotina, pois elas são independentes. Com isso, economiza-se um pouco de memória.

Também é interessante fazer com que a tabulação vertical seja controlada, de modo que se possam entrar valores com mais de 39 caracteres.



```
10  HOME
20  PRINT  TAB( 8)"DEMONSTRACAO
 DE ENTRADA", TAB( 55)"DE DADOS
"
30  VTAB 6: PRINT "SENHA DE ACE
SSO: ",:P$ = "G,05": GOSUB 600
 IF W$ < > "SENHA" THEN  END
40  VTAB 8: PRINT "DATA DE HOJE
 ",:P$ = "C" GOSUB 600 DT$ =
W$
50  VTAB 10  PRINT "HORA: ";:P$
 = "H": GOSUB 600:HR$ = W$
60  VTAB 12: PRINT "DIGITE UM V
ALOR: ": VTAB 13  HTAB 6 P$ = "
D10,3": GOSUB 600:VA =  VAL (W$
)
70  PRINT : PRINT DT$  PRINT HR
$: PRINT VA
80  END
596  REM  *********************
597  REM  *** SUB ENTRADA ***
598  REM  *********************
599  REM
600 PH =  PEEK (36) + 1:PP =  A
SC (P$)   66
601 WW$ = "_________________________
________________________": REM  * D
EFINA ESTA VARIAVEL NO INICIO D
O SEU PROGRAMA *
602  INVERSE
604  ON PP GOTO 606,608,0,0,612
,616
606 P = 8:PF = P:P1 = 2:P2 = 5:
WD$ = "DD/MM/AA":WC$ = "/": GOT
O 618
608 P =  VAL ( MID$ (P$,2,2)):D
 =  VAL ( RIGHT$ (P$,1)): IF D
= 0 THEN 614
610 PF = P:P1 = P - D - 1:P2 =
P1:WC$ = ".":WD$ =  LEFT$ (WW$,
P1) + WC$ +  LEFT$ (WW$,D): GOT
O 618
612 P    VAL ( RIGHT$ (P$,2))
614 PF = 1:P1 = 99:P2 = P1:WD$
=  LEFT$ (WW$,P):WC$   "": GOTO
 618
616 P = 5:PF = P:P1 - 2:P2 = P1
:WD$ = "HH:MM":WC$ = ":"
618  HTAB PH  PRINT WD$;
620 W$ = "":WI$ = ""
622 PX =  LEN (W$)
624  IF PX = P THEN 650
626 WA = 95 * (PP = 2 OR PP = 5
) + 68 * (PP = 1 AND PX < 2) +
77 * ((PP = 1 AND PX > 2 AND PX
 < 5) OR (PP = 6 AND PX > 2)) +
 65 * (PP = 1 AND PX > 5) + 72
* (PP = 6 AND PX < 2)
628  HTAB PH: PRINT  CHR$ (WA):
: HTAB PH: GET WI$
630 PA =  ASC (WI$): IF PA = 13
 AND PX >    PF THEN 650
632  IF PA = 8 AND PX > 0 THEN
644
634  IF PP = 5 THEN 638
636  IF PA < 48 OR PA > 57 THEN
 628
638  IF PA < 32 OR PA > 127 THE
N 628
640  HTAB PH: PRINT  CHR$ (PA);
:W$ = W$ + WI$:PX = PX + 1:PH =
 PH + 1: IF PX = P1 OR PX = P2
THEN PX = PX + 1:PH = PH + 1:W$
 = W$ + WC$
642  GOTO 624
644 PH = PH - 1: IF PX = 1 THEN
 620
646 PX = PX - 1: IF PX = P1 OR
PX = P2 THEN PH = PH - 1:PX = P
X - 1
648 W$ =  LEFT$ (W$,PX): GOTO 6
26
650  NORMAL : FOR X = 1 TO P -
 LEN (W$)
652  PRINT  CHR$ (32);: NEXT
654  ON PP GOTO 656,666,0,0,666
,660
656 D =  VAL ( LEFT$ (W$,2)):M
=  VAL ( MID$ (W$,4,2)):A =  VA
L ( RIGHT$ (W$,2))
658 V = ((M = 4) + (M = 6) + (M
 = 9) + (M = 11)) * (D > 30) +
(M = 2) * (( INT (A / 4) = A /
4) * (D > 29) + ( INT (A / 4) <
 > (A / 4)) * (D > 28)) + ((A
< 1) + (M < 1) + (M > 12) + (D
< 1) + (D > 31)): GOTO 664
660 H =  VAL ( LEFT$ (W$,2)):M
=  VAL ( RIGHT$ (W$,2))
662 V = (H > 24) + (M > 59)
664  IF V THEN  PRINT  CHR$ (7)
;:PH = PH -  LEN (W$): INVERSE
: GOTO 618
666  PRINT : RETURN
```

UM EDITOR MUSICAL (1)

Transforme o microcomputador em uma partitura com composições musicais de sua própria autoria e, ainda, em uma orquestra que execute as melodias recém-criadas.

Existem duas maneiras de fazer o microcomputador tocar música. A primeira consiste em introduzir a melodia codificada em números dentro de um programa. A segunda, em utilizar um editor musical que cuide de toda a codificação, deixando ao usuário apenas a tarefa de digitar as notas.

Já vimos em *INPUT* como programar o micro para executar melodias. O processo é simples, mas requer um bom conhecimento dos comandos musicais do BASIC. Como estes nem sempre têm muita relação com música, o "compositor" é obrigado a consultar constantemente tabelas ou manuais, para verificar os valores adequados para a tonalidade, duração e volume, entre outros parâmetros de cada nota. A notação macromusical do MSX e do TRS-Color facilita esse processo, mas não elimina o trabalho com o restante do programa.

A execução de música através de programas apresenta outras desvantagens, como a dificuldade de editar a melodia e a impossibilidade de ouvir as notas antes que o programa esteja completo. Além disso, para cada peça temos que escrever um novo programa.

Um editor musical cuida de todos os detalhes da codificação, deixando o usuário livre para tratar da melodia propriamente dita. Não é preciso conhecimentos de programação para usá-lo.

Como os programas são bem longos, foram divididos em três partes. Digite agora a primeira parte e grave-a, para mais tarde completar a listagem.

COMO DIGITAR AS NOTAS

Uma vez que tenhamos o programa completo na memória do computador, poderemos selecionar as notas de várias maneiras. O processo será explicado pormenorizadamente para cada uma das máquinas no último artigo desta série.

Seja qual for o método escolhido, o usuário sempre terá a chance de alterar a melodia — mudando, inserindo ou apagando notas — até obter o resultado desejado. Se quiser conferir o trabalho, poderá executar a música — mesmo incompleta — a qualquer instante.

O editor musical permite ainda a mudança do andamento da oitava e, no caso do MSX, do timbre da nota.

As notas são selecionadas por meio de seus nomes (A, B, C, D...) ou do teclado, que é usado como se fosse um piano. Se o usuário preferir, poderá combinar os dois sistemas, introduzindo nota por nota em certos trechos e executando outros no teclado.

S

```
10 BORDER 0: PAPER 0: INK 7:
CLS
40 LET maxnotes=1500: LET ct=
0: LET tempo=.1: POKE 23609,
128: POKE 23658,0: DIM m(35)
95 FOR i=1 TO 35: READ m(i):
NEXT i
100 DIM t(maxnotes+1): GOTO
110
105 CLS : PRINT "Aguarde. Musi
ca sendo apagada."
110 FOR i=1 TO maxnotes: LET t
(i)=0: NEXT i
190 CLS
200 PRINT  INVERSE 1;AT 0,6;"
MENU PRINCIPAL ": INVERSE 0;''
'TAB 5,"[1] Tocar pelo teclado
"''TAB 5,"[2] Entrar  notas"''
TAB 5,"[3] Tocar de novo"
210 PRINT 'TAB 5;"[4] Editar m
usica"''TAB 5;"[5] Apagar musi
ca"''TAB 5;"[6] Salvar musica"
''TAB 5;"[7] Carregar musica"
270 PRINT ''TAB 5;"Opcao [S] p
ara sair";
300 LET A$=INKEY$: IF A$=""
THEN  GOTO 300
310 LET A=CODE (A$)
320 IF (A<49 OR A>55) AND A<>
113 AND A<>81 THEN  GOTO 190
330 IF A=49 THEN  GOSUB 1000
340 IF A=50 THEN  GOSUB 2000
350 IF A=51 THEN  GOSUB 3000
360 IF A=52 THEN  GOSUB 4000
365 IF A=53 THEN  GOTO 105
366 IF A=54 THEN  GOSUB 5000
367 IF A=55 THEN  GOSUB 6000
370 IF A=115 OR A=83 THEN
STOP
380 GOTO 190
1000 CLS
1001 LET len=1: LET I$="Semicol
cheia"
1002 INPUT "[C]ONTINUE DA ULTIM
A NOTA    OU [I]NICIE NOVA MUSI
CA ? ";i$
1003 IF i$<>"c" AND i$<>"i" THE
N  GOTO 1002
1004 LET num=1: IF i$="c" THEN
 LET num=ct*2+1: LET tn=num
1005 IF i$="i" THEN  LET ct=0
1010 PRINT "Toque as notas nas
teclas de"'"<Q> - mais baixa
   ate"'"<M> - mais alta"'"Duas
 oitavas e meia estao dispo-niv
eis, sendo a tecla <I> o     C
medio"''''
1060 PRINT AT 17,0;"Duracao da
nota="
1070 FOR i=1 TO 100: NEXT i
1075 LET O$=""
1080 LET N$=INKEY$: IF N$="" TH
EN  GOTO 1075
```



- EXECUCÃO DE MELODIAS ATRAVÉS DE UM PROGRAMA
- VANTAGENS DO EDITOR MUSICAL
- COMO DIGITAR
- AS NOTAS
- ALTERACÃO DA MELODIA
- CONFERÊNCIA
- COMPOSICÃO DE PECAS
- SELECÃO DE NOTAS

```
1085 IF N$=O$ THEN  GOTO 1080
1090 LET N=CODE (N$)
1091 IF N=33 THEN  LET len=1: L
ET I$="Semicolcheia"
1092 IF N=64 THEN  LET len=2: L
ET I$="Colcheia"
1093 IF N=35 THEN  LET len=4: L
ET I$="Seminima"
1094 IF N=36 THEN  LET len=8: L
ET I$="Minima"
1096 IF N=37 THEN  LET len=16:
LET I$="Semibreve"
1097 PRINT AT 17,16,"  ",I$;"
  "
1099 IF N<>32 AND N<>13 THEN  G
OTO 1109
1100 IF i$="i" THEN  LET ct=INT
 (num/2)· RETURN
1102 LET ct=ct+INT ((num-tn)/2)
: RETURN
1109 IF N<60 THEN  LET index=N·
47
1110 IF N>90 THEN  LET index=N-
87
1112 IF N>=60 AND N<=90 THEN  G
OTO 1080
1115 IF index=5 OR index=9 OR i
ndex=16 OR index=20 OR index 21
 THEN  GOTO 1080
1116 IF index=2 THEN  LET t(num
)=len: LET t(num+1)=-4· GOTO 11
27
1118 IF index<=0 THEN  GOTO 108
0
1120 SOUND len/10,m(index)
1125 LET t(num) len· LET t(num+
1)=m(index)
1126 PRINT AT 15,0;INT (num/2)+
1;" notas na memoria"
1127 LET num=num+2
1130 LET O$=N$
1140 GOTO 1080
```



```
10 CLEAR 5000·MX=250
20 R1$="whqestu"·R2$="12345678"
30 R3$="abcdefgCDFGAp"
40 NY$="Q2W3ER5T6Y7UI900PZSXDCF
VBHNJM,L.Ç/[<"+CHR$(30)+CHR$(31
)+CHR$(29)+CHR$(28)+" "+CHR$(18
)+CHR$(127)+CHR$(13)+CHR$(11)
50 DIM N$(MX),C$(12)
60 FOR I=1TO7:READ LE$(I),L2$(I
):NEXT
70 FOR I=1TO12 ·READ C$(I)·NEXT
80 DATA SEMIBREVE,1,MINIMA,2,SE
MINIMA,4,COLCHEIA,8,SEMICOLCHEI
A,16,FUSA,32,SEMIFUSA,64
90 DATA c,C,d,D,e,f,F,g,G,a,A,b
100 FOR I=1 TO MX·N$(I)="****"·
NEXT
110 NN=0· OC$="4"·TE=120·LE$="w
"  [E=1
120 CLS·COLOR 15,2·KEY OFF·LOCA
TE 4,1:PRINT"COMPOSITOR MUSICAL
 MENU PRINCIPAL"
130 LOCATE 6,3·PRINT"1:RECUPERA
 MUSICA DA FITA"
140 LOCATE 6,4:PRINT"2:SALVA MU
SICA NA FITA"
150 LOCATE 6,5·PRINT"3·TOQUE NO
 TECLADO"
160 LOCATE 6,6·PRINT"4·COLOQUE
UMA TABELA DE NOTAS"
170 LOCATE 6,7:PRINT"5·MUDE O T
EMPO DE EXECUÇÃO"
180 LOCATE 6,8·PRINT"6·LISTA/ED
ITA NOTAS"
190 LOCATE 6,9:PRINT"7:TOCA A M
USICA DA MEMORIA"
200 LOCATE 6,10 PRINT"8·MUDANÇA
 GERAL DE OITAVA"
210 LOCATE 6,11·PRINT"9·SAI DO
PROGRAMA"
220 LOCATE 5,19:PRINT"ENTRE COM
 O NUMERO DA OPÇAO":
230 A$=INKEY$ ·IF A$<"1" OR A$>
"9" THEN230
```

```
240 OP=VAL(AS)
250 ON OP GOSUB 2080,2210,500,8
10,300,1510,1760,1910,270
260 GOTO 120
270 CLS:PRINT"VOCE TEM CERTEZA(
S/N)?"
280 A$=INKEY$:IF A$<>"S" AND A$
<>"N" THEN 280
290 IF A$="S" THEN CLS·COLOR 15
,4:END ELSE 120
300 CLS COLOR 15,4:PRINT"MUDANÇ
A DO TEMPO DE EXECUÇAO"
310 PRINT:PRINT"TEMPO ATUAL:";T
E
320 PRINT"ENTRE NOVO VALOR DE T
EMPO";·INPUT ST
330 IF ST=0 THEN RETURN
340 IF ST<32 OR ST>255 THEN 320
350 TE=ST·RETURN
360 LOCATE 6,7·PRINT"NOTA     OI
TAVA     DURAÇAO"
370 U=9
380 FOR L=NN-5TONN
390 IF L<1 THEN 490
400 LOCATE 1,U:PRINT TAB(20);"
                    ";
410 LOCATE 1,U:PRINT USING"###
";L,
420 A$=NS(L)
430 B$=LEFT$(A$,1)·IF B$>="a" A
ND B$<="g" THEN C$=CHR$(ASC(B$)
-32)+" "ELSE C$=B$+"#"
440 IF B$="p" THEN C$="- "
450 PRINT "  ";C$;"         ";MID
$(A$,2,1),"          ";
460 PRINT LE$(INSTR(R1$,MID$(A$
,3,1)));
470 IF MID$(A$,4,1)="." THEN PR
INT"." ELSE PRINT
480 U=U+1:
490 NEXTL·RETURN
500 CLS·COLOR 15,4,15·C=VAL(OC$
):IF C>6 THEN C=6:OC$="6"
510 LOCATE 13,1:PRINT"MODO ORGA
O"
520 LOCATE 16,4:PRINT ;"
   ";
530 LOCATE 1,3·PRINT"OITAVA MAX
IMA·";C;TAB(46);"DURAÇAO:";LE$(
INSTR(R1$,LEFT$(LE$,1)));
540 IF MID$(LE$,2,1)="." THEN P
RINT"." ELSE PRINT
550 LOCATE 0,19·PRINT"---------
---------------------------- "
;
560 LOCATE 0,20·PRINT" /  =OITA
VA       (INS/DEL) /  =DURAÇAO"
;
570 LOCATE 0,21:PRINT"ENTER =ME
NU                  CLS=APAGA"
;
580 GOSUB 360
590 IF NN=MX THEN LOCATE 1,18·P
RINT"MAXIMO NUMERO DE NOTAS COL
OCADO!";·FOR S=1TO3000·NEXT:RET
URN
600 I$=INKEY$·IF I$="" THEN 600
610 P=INSTR(NY$,I$)·IF P=0 THEN
 600
620 IF P>36 THEN 650
630 OC$=MID$(STR$(C+INT((P-1)/1
2)),2)·I$=C$(P-12*INT((P-1)/12)
)
640 GOTO 750
```



```
10 CLEAR 5000
20 MX=250:R1$="whqes":R2$="1234
5":R3$="abcdefgCDFGAp":NY$="Q2W
3ER5T6Y7UI900P@AZSXDCVGBHNMK,L.
"+CHR$(103)+"/^"+CHR$(10)+CHR$(
8)+CHR$(9)+" "+CHR$(21)+CHR$(93
)+CHR$(13)+CHR$(12)
30 DIM NS(MX),C$(12)
40 FOR I=1 TO 5:READ LE$(I),L2$
(I):NEXT
50 FOR I=1 TO 12:READ C$(I):NEX
T
60 DATA SEMIBREVE,1,MINIMA,2,SE
MINIMA,4,COLCHEIA,8,SEMICOLCHEI
A,16
70 DATA c,C,d,D,e,f,F,g,G,a,A,b
80 FOR I=1 TO MX:NS(I)="*****":N
EXT
90 NN=0:OC$="3":TE=8:LE$="w ":L
E=1
100 CLS:PRINT@6,"COMPOSITOR MUS
ICAL":PRINT @40,"MENU PRINCIPAL
"
110 PRINT@96,"1:RECUPERA MUSICA
 DA FITA"
120 PRINT"2:SALVA MUSICA ATUAL
NA FITA"
130 PRINT"3:TOQUE NO TECLADO"
140 PRINT"4:COLOQUE UMA TABELA
DE NOTAS"
150 PRINT"5:MUDE O TEMPO DE DUR
ACAO"
160 PRINT"6.LISTA/EDITA NOTAS"
170 PRINT"7:TOCA A MUSICA DA ME
MORIA"
180 PRINT"8:MUDA A OITAVA GERAL
"
190 PRINT"9:SAI DO PROGRAMA"
200 PRINT@452,"ENTRE O NUMERO D
A OPCAO";
210 A$=INKEY$:IF A$<"1" OR A$>"
9" THEN 210
220 OP=VAL(A$)
230 ON OP GOSUB 1900,2020,480,7
60,420,1360,1560,1720,250
240 GOTO 100
250 CLS:PRINT" VOCE TEM CERTEZA
(S/N)?":POKE 282,255
260 A$=INKEY$:IF A$<>"S" AND A$
<>"N" THEN 260
270 IF A$="S" THEN CLS:END ELSE
100
280 PRINT@132,"NOTA  OITAVA   D
URACAO"
290 RT=0
300 FOR L=160 TO 320 STEP 32:PR
INT@L:NEXTL:PRINT@160,"";
310 FOR L=NN-5TO NN
320 IF L<1 THEN 410
330 PRINTUSING"###  ";L;
340 A$=NS(L)
350 B$=LEFT$(A$,1):IF B$>="a"AN
D B$<="g" THEN C$=CHR$(ASC(B$)-
32)+" "ELSE C$=B$+"#"
360 IF B$="p" THEN C$="- "
370 PRINT C$;"       ";MID$(A$,2,
1),"      ";
380 PRINT LE$(INSTR(R1$,MID$(A$
,3,1)));
```

MICRO DICAS

MELHORE A QUALIDADE DO SOM

O programa apresentado neste artigo é bastante satisfatório para a composição e execução musical a nível amador. Os músicos profissionais, entretanto, logo notarão suas limitações — entre elas, a falta de recursos para a modulação do timbre (o que diferencia os vários instrumentos entre si quanto à sonoridade) e a impossibilidade de se obter efeitos polifônicos. Mas a culpa não é do programa

Mesmo o MSX, o microcomputador mais sofisticado em termos de programação de efeitos sonoros, não é capaz de substituir um verdadeiro sintetizador musical. Este permite a simulação de um grande número de instrumentos de sopro, corda ou percussão, através da manipulação das formas de onda. Acoplando-o a seu microcomputador, por meio de uma interface MIDI (veja o artigo da página 1306), você poderá conseguir a qualidade máxima em matéria de som.

```
390 IF MID$(A$,4,1)=" "THEN PRI
NT"."ELSE PRINT
400 IF RT<>0THEN RETURN
410 NEXT L:RETURN
420 CLS:PRINT@7,"MUDE OPCAO TEM
PO"
430 PRINT:PRINT"TEMPO ATUAL E";
TE
440 PRINT@128,"ENTRE NOVO VALOR
 DE TEMPO";:INPUT ST
450 IF ST=0 THEN RETURN
460 IF ST<0ORST>255 THEN 440
470 TE=ST·RETURN
480 CLS:POKE282,255·C=VAL(OC$):
IF C>3 THEN C=3·OC$="3"
490 PRINT@7,"MODO ORGAO"
500 PRINT@64,"OITAVA ":C;TAB(10
),"DURACAO:";LE$(INSTR(R1$,LEFT
$(LE$,1)));
510 IF MID$(LE$,2,1)="." THEN P
RINT"." ELSE PRINT
520 PRINT@448,"C/B=OITAVA
ESQ/DR=DURACAO   ENTER=MENU
  CLEAR=APAGAR";
530 GOSUB 280
540 IF NN=MX THEN PRINT@416,"MA
XIMO NUMERO DE NOTAS COLOCADO!"
;:FORD=1TO1000:NEXT:RETURN
550 I$=INKEY$ IF I$="" THEN 550
560 P=INSTR(NY$,I$):IF P=0 THEN
 550
570 IF P>36 THEN 600
580 OC$=MID$(STR$(C+INT((P-1)/1
2)),2)·I$=C$(P 12*INT((P-1)/12)
)
590 GOTO 700
```

MAIS OPERAÇÕES COM CADEIAS

- READ E DATA
- MENUS E OPÇÕES
- RECUPERAÇÃO DE PALAVRAS
- O USO DE MACROS
- A INSTRUÇÃO CLEAR

Apresentamos neste artigo algumas funções e rotinas relativamente simples, que podem facilitar bastante as tarefas que envolvem a manipulação de cadeias de caracteres. Conheça-as.

A maneira mais adequada de armazenar conjuntos de dados numéricos ou alfanuméricos dentro de um programa em BASIC consiste em colocá-los em linhas **DATA**, e lê-los com comandos **READ**.

Porém, quando o número de dados é pequeno, não compensa introduzi-los em linhas **DATA** e acrescentar ao programa comandos destinados a lê-los e carregá-los em variáveis indexadas. É muito mais conveniente colocar os dados dentro de uma única variável *string*, de tal forma que possamos recuperá-los ou consultá-los por meio de uma subrotina ou função.

Esta é também a melhor solução para os usuários de computadores que não dispõem dos comandos **READ** e **DATA**, como os micros pessoais ZX-81.

TRINTA DIAS TEM SETEMBRO...

O programa que se segue exemplifica bem o procedimento sugerido. Ele informa quantos dias tem um determinado mês do ano:



```
20 LET C$="312931303130313130313031"
40 PRINT "ENTRE O NUMERO DO MES ";
45 INPUT M
50 IF M<1 OR M>12 THEN GOTO 40
60 PRINT "ESTE MES TEM ";
70 PRINT C$((M-1)*2+1 TO (M-1)*2+2)," DIAS."
80 GOTO 40
```



```
20 C$="312931303130313130313031"
40 INPUT "ENTRE O NUMERO DO MES ";M
50 IF M<1 OR M>12 THEN GOTO 40
60 PRINT "ESTE MES TEM ";
70 PRINT MID$(C$,(M-1)*2+1,2);"DIAS."
80 GOTO 40
```

Os números de dias referentes a cada mês são colocados na cadeia de caracteres **C$**. A função da linha 70 extrai o número de dias do mês **M** (especificado no programa de teste, pela entrada da linha 40) usando a expressão:

posição inicial em **M$** = (M – 1)*2 + 1
número de dígitos = 2

Note que, se **M** = **1**, a posição inicial em **M$** é 1. Esta é uma fórmula geral, que pode ser usada para extrair subcadeias de comprimento fixo, concatenadas em uma única cadeia.

Nos computadores que têm funções programáveis do tipo string, essa fórmula pode ser colocada dentro de uma função, e usada repetidas vezes, como neste programa:



```
10 DEF FNT$(I,L,S$)=MID$(S$,(I-1)*L+1,L)
20 LET M$="JANFEVMARABRMAIJUNJULAGOSETOUTNOVDEZ"
30 LET C$="312931303130313130313031"
40 INPUT "ENTRE O NUMERO DO MES ";M
50 IF M<1 OR M>12 THEN GOTO 80
60 PRINT FNT$(M,3,M$);" = ";FNT$(M,2,C$);" DIAS."
70 GOTO 40
80 PRINT:PRINT "DATA INVALIDA":GOTO 40
```



Para executar o programa no Spectrum, substitua a linha 10:

```
10 DEF FNT$(I,L,S$)=S$((I-1)*L+1 TO (I-1)*L+L)
```

Na função **FNT$**, definida na linha 10, o primeiro argumento é o índice; o segundo, o tamanho fixo das subcadeias e o terceiro, o nome da cadeia com os dados. Note que a mesma função é usada com duas cadeias de formatos diferentes, na linha 60 do programa.

Em computadores que não aceitam funções com mais de um argumento, pode-se empregar uma sub-rotina, mais longa, mas igualmente útil:



```
20 LET M$="JANFEVMARABRMAIJUNJULAGOSETOUTNOVDEZ"
30 LET C$="312931303130313130313031"
40 INPUT "ENTRE O NUMERO DO MES ";M
50 IF M<1 OR M>12 THEN GOTO 98
55 LET I=M
60 LET L=3
65 LET X$=M$
70 GOSUB 100
75 PRINT S$;" TEM ";
80 LET X$=C$
85 GOSUB 100
90 PRINT S$;" DIAS."
95 GOTO 40
98 PRINT:PRINT "DATA INVALIDA"
99 GOTO 40
100 LET S$=MID$(X$,(I-1)*L+1,L)
110 RETURN
```



Para rodar o programa nos micros Spectrum e ZX-81, substitua a linha 100 do programa:

```
100 LET S$=X$((I-1)*L+1 TO (I-1)*L+L)
```

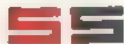
MENUS E OPÇÕES

Outra aplicação bastante interessante dos conceitos aqui discutidos consiste na programação rápida de menus com entradas por extenso. As várias opções, colocadas em uma única variável string, são depois recuperadas pelo método já descrito:



```
10 LET FL$="INSEREEDITARAPAGARLISTAR"
20 PRINT "ENTRE UM COMANDO: ";F
```

```
L$;" ";
30 INPUT OP$
40 FOR I=1 TO 4
50 LET D$=MID$(FL$,(I-1)*6+1,6)
60 IF D$=OP$ THEN GOTO 80
65 NEXT I
70 PRINT "CODIGO INVALIDO"
75 GOTO 20
80 PRINT "COMANDO ESCOLHIDO =>
";D$
90 GOTO 20
```



Substitua a seguinte linha do programa para micros da linha Sinclair:

```
50 LET D$=FL$((I-1)*6+1 TO (I-1
)*6+6)
```

Para os micros que dispõem da instrução **INSTR**, que localiza subcadeias, e de funções programáveis, podemos fazer um programa mais compacto:



```
5 DEF FNC(O$,C$,N)=INT((INSTR(O
$,LEFT$(C$+STRING$(N," "),N))-1
)/N)+1
10 O$="CR CH PD DB AT "
20 PRINT "ENTRE O CODIGO DA TRA
NSACAO ";
30 PRINT "(CODIGOS VALIDOS: ";O
$;")"
40 LINE INPUT "CODIGO : ";C$
50 I=FNC(O$,C$,3)
60 IF I=0 THEN PRINT "*** CODIG
O INVALIDO":GOTO 30
70 PRINT "CODIGO ";I
80 GOTO 20
```

A função **FNC$** retorna o número da opção (posição seqüencial na cadeia). Seus argumentos de entrada são:

O$: cadeia que contém as opções;
C$**: subcadeia procurada em **O$;
N: tamanho fixo das subcadeias.

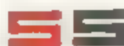
É necessário colocar espaços em branco para separar os comandos armazenados em **O$**. Se isso não for feito, a função **INSTR** pode encontrar subcadeias superpostas, que não correspondem às opções possíveis.

RECUPERAÇÃO DE PALAVRAS

A recuperação de palavras, códigos ou números de uma lista, efetuada de acordo com o método que acabamos de explicar, requer que todos eles tenham o mesmo comprimento, de modo a permitir a aplicação de uma fórmula.

Isso nem sempre é desejável ou possível. Suponhamos que, na programação de um jogo de aventura, você peça ao jogador que entre a lista dos objetos que ele quer levar em uma incursão a um castelo mal-assombrado. Nesse caso, os comprimentos das palavras contidas na cadeia de entrada podem ser bastante diferentes entre si, o que exigiria a elaboração de um programa um pouco mais "inteligente".

Se determinarmos, por exemplo, que o caractere de separação entre palavras é o espaço em branco, a seguinte rotina realizará o trabalho:



```
80 REM  ---- SUBROTINA  -------
100 LET A$=""
110 IF C$$(1)="" THEN RETURN
120 IF C$$(1)<>" " THEN GOTO 13
0
125 LET C$$=C$$(2 TO)
127 GOTO 100
130 LET A$=A$+C$$(1)
135 LET C$$=C$$(2 TO)
140 IF C$$(1)="" OR C$$(1)=" "
 THEN RETURN
150 GOTO 130
```



```
80 REM ----- SUBROTINA  ~   ---
100 A$=""
110 IF MID$(C$$,1,1)="" THEN RE
TURN
120 IF MID$(C$$,1,1)=" " THEN C
$$=MID$(C$$,2):GOTO 100
130 A$=A$+MID$(C$$,1,1):C$$=MID
$(C$$,2)
140 IF MID$(C$$,1,1)="" OR MID$
(C$$,1,1)=" " THEN RETURN
150 GOTO 130
```

A cadeia de entrada é **C$$**. A cada chamada da rotina, **A$** retorna uma subcadeia contida entre dois espaços em branco, em **C$$**; extraindo-a, reduz o tamanho de **C$$**.

Eis um programa que testa o funcionamento dessa rotina:



```
20 PRINT "ENTRE TRES COISAS QUE
 CARREGARA  (SEPARADAS POR ESPA
COS):"
30 INPUT C$$
40 LET N=0
50 PRINT"VOCE CARREGARA:"
55 GOSUB 100
60 IF A$="" THEN GOTO 80
70 LET N=N+1:PRINT N,A$
75 GOTO 60
80 PRINT "E MAIS NADA..."
90 STOP
```



Para rodar nesses computadores, acrescente a seguinte linha:

```
10 CLEAR 1000
```

A linha 100 inicializa a cadeia de saída, **A$**. A linha 110 começa um laço de repetição, testando se a cadeia de entrada, **C$$**, é vazia — ou seja, se todas as palavras nela contidas já foram extraídas. Em caso afirmativo, a sub-rotina retorna o controle ao programa que a chamou. A linha 120 verifica se o primeiro dos caracteres que restaram em **C$$** é um espaço em branco. Se for, ele é "podado" de **C$$**, e o programa retorna para a linha 100, para executar um novo teste. Se o primeiro caractere não for um espaço em branco, a linha 130 retira-o de **C$$**, levando-o para **A$**. Finalmente, se um novo espaço em branco for encontrado (fim de subcadeia), ou se **C$$** tiver se esgotado, a linha 140 retorna o controle ao seu programa principal.

SUBSTITUIÇÃO DE SUBCADEIAS

Uma das funções mais úteis de um programa de processamento de textos é a que permite substituir todas as ocorrências de uma determinada palavra, ou seqüência de caracteres, em um documento. Essa função possibilita, entre outras coisas, o uso de *macros* para a redação rápida de textos muito repetitivos. Digamos que você esteja escrevendo um artigo sobre processamento de imagens, e que essa expressão ("processamento de imagens") será citada algumas dezenas de vezes, por extenso, ao longo do texto. Para poupar tempo e esforço, sempre que for preciso digitá-la, você poderá colocar um símbolo qualquer em seu lugar — por exemplo, dois caracteres que dificilmente aparecerão juntos em outras partes do texto: *P, $$ etc. Depois, bastará acionar a função de busca e substituição, e o programa trocará o símbolo convencionado pela expressão por extenso, sempre que o encontrar.

Programar essa operação em BASIC não chega a ser difícil, mas envolve algumas complicações, pois a sequência que irá substituir o símbolo no texto pode ter um número de caracteres menor, igual ou maior. Além disso, a maioria dos microcomputadores não aceita variáveis string, com mais de 255 caracteres. Assim, durante o processo de substituição, é necessário fazer com que o programa verifique esse número, para

não ocorrer nenhum erro. Apresentamos, a seguir, uma sub-rotina de busca e substituição, que você poderá adicionar a seus programas:

```
100 LET N=1
110 IF LEN A$ -LEN B$ +LEN C$ >
255 THEN RETURN
120 FOR I=N TO LEN A$
130 IF A$(I TO I+LEN B$-1)=B$ T
HEN GOTO 160
140 NEXT I
150 RETURN
160 LET A$=A$(TO I-1)+C$+A$(I+L
EN B$ TO)
170 LET N=I+LEN C$
180 GOTO 110
```

```
100 N=1
110 IF LEN(A$)-LEN(B$)+LEN(C$)>
255 THEN RETURN
120 FOR I=N TO LEN(A$)
130 IF MID$(A$,I,LEN(B$))=B$ TH
EN GOTO 160
140 NEXT I:RETURN
160 A$=LEFT$(A$,I-1)+C$+MID$(A$
,I+LEN(B$))
170 N=I+LEN(C$)
180 GOTO 110
```

Os argumentos de entrada da subrotina são:
A$ - cadeia que terá substituições;
B$** - subcadeia em **A$ a substituir;
C$** - subcadeia que substituirá **B$.

Suponhamos que a célebre frase de Gertrude Stein:

UMA ROSA E' UMA ROSA E' UMA ROSA

deverá ser transformada em outra (não menos poética):

UMA PEDRA E' UMA PEDRA E' UMA PEDRA

Neste exemplo, então:

A$ = UMA ROSA E' UMA ROSA E' UMA ROSA
B$ = ROSA
C$ = PEDRA

A rotina funciona da seguinte maneira: a linha 100 inicializa **N** em 1. Essa variável aponta o caractere de **A$** a partir do qual **B$** será procurado. Na linha 110, checamos se o comprimento total da cadeia substituída não excede 255. O laço que vai da linha 120 à 140 verifica se **B$** está presente em **A$**, a partir do caractere **N** até o fim. Se estiver, a linha 160 efetua a substituição, concatenando a primeira seção de **A$**, até a ocorrência de **B$**, mais **C$** e a parte restante de **A$**. **N** é então atualizado e ocorre nova busca por **B$** (retorno à linha 110).

Nos micros com função **INSTR** (*instring*), o laço das linhas 120 a 140 pode ser substituído por uma só linha:

```
120 I=INSTR(I,A$,B$):IF I=0 THE
N RETURN
```

Esta listagem testa a sub-rotina de substituição. Ela é auto-explicativa:

```
20 PRINT "ENTRE UMA FRASE ";
25 INPUT A$
30 PRINT "PALAVRA PARA SUBSTITU
IR ";
35 INPUT B$
40 PRINT "POR ";
45 INPUT C$
50 GOSUB 100
60 PRINT A$
70 GOTO 20
```

Para rodar o programa no TRS-80 e no TRS-Color, adicione esta linha:

```
10 CLEAR 1000
```

Se você quiser substituir todas as ocorrências de uma subcadeia, em um texto composto de várias linhas (armazenadas em **DATA**, por exemplo), utilize o próximo programa. Não se esqueça de colocar uma última linha **DATA** "*" para assinalar o fim do texto:

```
10 INPUT "PALAVRA PARA SUBSTITU
IR ";
20 INPUT "POR ";
30 READ A$
40 IF A$="*" THEN GOTO 70
50 GOSUB 70:PRINT A$
60 GOTO 30
70 STOP
```

A INSTRUÇÃO CLEAR

A programação de variáveis string apresenta algumas diferenças conforme o tipo de interpretador BASIC usado.

Como vimos no artigo anterior desta série (página 1214), os interpretadores se dividem em dois grandes grupos quanto às funções de tratamento de variáveis literais. No primeiro grupo, chamado Microsoft (nome da *software house* norte-americana responsável pelo desenvolvimento do BASIC padrão para os micros MSX, TK-2000, Apple II, TRS-80 e TRS-Color, entre outros), encontramos funções familiares como **LEFT$**, **RIGHT$**, **MID$**, **INSTR** etc.

No segundo grupo, denominado Sinclair (destinado às máquinas compatíveis com as linhas ZX-81 e Spectrum), a cadeia de caracteres é tratada como um conjunto dimensionado, e a partícula **TO** se encarrega da referência às subcadeias de um string.

Os interpretadores do tipo Microsoft também diferem dos do tipo Sinclair quanto ao gerenciamento do espaço disponível na memória RAM para variáveis literais e numéricas. Ambos, porém, armazenam uma variável literal de forma semelhante (veja o artigo da página 1101): uma cadeia pode ter entre 0 e 255 caracteres, armazenados em um conjunto contíguo de bytes. O primeiro byte de um string é usado para armazenar o seu comprimento, ou seja, o número de caracteres que possui. Essa informação é usada, por exemplo, pela função **LEN**, disponível nos dois tipos de interpretador, que informa o comprimento da cadeia. Se uma variável string for concatenada através de uma expressão como **LET A$ = A$ + B$**, o interpretador manipula **A$** e **B$** de modo a transferi-los para uma área livre da memória RAM, colocá-los um após o outro, eliminar o byte de comprimento do segundo string (**B$**) e atualizar o valor armazenado no byte de comprimento do primeiro string (**A$**).

Como se pode concluir, as operações de concatenação são lentas, devendo ser evitadas quando possível. Além disso, exigem uma área livre de memória para a execução da cópia — os originais dos dois strings **A$** e **B$** continuam no mesmo lugar, mas "desativados" pelo interpretador. Periodicamente, uma função chamada "coleta de lixo" (*garbage collection*, em inglês) recupera essas áreas para uso pelo programa.

Quanto à reserva de espaço para strings, na RAM, os interpretadores também apresentam diferenças. Os do tipo Sinclair operam automaticamente ou por meio de uma declaração **DIM**, que deve ser usada para dimensionar strings. Os do tipo Microsoft nem sempre precisam de uma declaração explícita para dimensionar o tamanho da área de strings. Nos micros das linhas TRS-80 e TRS-Color, a instrução **CLEAR** serve a esse propósito. No Apple e no MSX, ela não é necessária.

TROCA DE MENSAGENS

Compartilhe informações profissionais e de lazer com outras pessoas através de seu micro. Basta ligá-lo por telefone a um dos vários serviços computadorizados de "quadro de avisos"

No artigo *Sua ligação com o mundo* (página 561), examinamos como o dispositivo especial chamado *modem* pode conectar um micro a outros computadores, utilizando a linha telefônica. Através dele, uma grande variedade de bancos de dados computadorizados ficará à disposição do usuário: cotações das Bolsas, noticiários, referências bibliográficas sobre assuntos técnicos etc. Existem hoje, em todo o mundo, mais de um milhão de pessoas que se beneficiam de redes de computadores.

O acesso a esses sistemas, entretanto, não é algo exatamente barato, principalmente quando é necessário pagar ligações internacionais (DDI). Além disso, muitos usuários se ressentem da falta de um contato mais pessoal entre quem fornece e quem recebe a informação — o que talvez explique, por exemplo, o sucesso e a popularidade do radioamadorismo. Contudo, já existe um equivalente desse hobby dentro do campo das redes informatizadas de dados. Trata-se da versão "doméstica" dos grandes sistemas telemáticos (como CompuServe e Cirandão) denominada *quadro de avisos computadorizado* (do inglês, *bulletin board*).

Um *Computerized Bulletin Board Service* (ou CBBS, como é mais conhecido entre os aficionados) é, em sua forma mais simples, o equivalente eletrônico do quadro de avisos de um clube, onde todos podem afixar notícias, recados etc. Para ter acesso a um sistema desse tipo, o usuário precisa — além de um telefone e de um micro — dos seguintes elementos: um modem, um software específico para intercomunicação e o número do telefone de um CBBS. O custo do acesso propriamente dito (ler e/ou escrever no quadro de avisos), geralmente muito baixo, varia segundo a finalidade do serviço (comercial ou não) e inclui a despesa normal com a ligação telefônica.

Embora os CBBS existam há algum tempo (sobretudo nos EUA, onde foram criados), em nosso país eles ainda são uma novidade. Apenas as cidades maiores, como São Paulo e Rio de Janeiro, dispõem desse serviço. Tudo indica, entretanto, que os CBBS se multiplicarão rapidamente e que a maioria dos proprietários de modems tomará contato com eles brevemente.

CARREGAR E DESCARREGAR

Os CBBS não são utilizados somente como sistemas de mensagens. Na realidade, sua rápida expansão se deve, principalmente, ao fato de a maioria deles dispor de um serviço especial que possibilita a transmissão de software de um computador para outro. No jargão dos seus aficionados, *uploading* significa o ato de enviar um software para armazenamento no CBBS, enquanto *downloading* quer dizer retirar uma cópia das informações disponíveis. É importante ressaltar aqui que o micro que manda e o que recebe a listagem do programa não precisam ser compatíveis entre si. Essa é, portanto, uma oportunidade rara de transportar certos tipos de software por meios incomuns.

O princípio dessa idéia consiste em converter os programas em textos ASCII (padronizados para todos os micros, que abordamos, menos o ZX-81) para realizar a transmissão via modem.

Outro requisito para o trabalho de copiar os programas disponíveis no CBBS é ter uma unidade de discos e um interpretador ou compilador da linguagem original do programa. Com eles, poderemos carregar arquivos em formato ASCII. Os sistemas operacionais das linhas TRS-80, TRS-Color e MSX oferecem essa facilidade sem requerer o emprego de programas conversores especiais. Em outros micros, o próprio software de transmissão pode incluir uma pequena rotina que transforma texto em programas e vice-versa. Além disso, alguns CBBS são específicos para uma linha de micros (os da linha Apple e IBM PC são os mais comuns), e permitem a transferência direta de programas entre dois computadores compatíveis, sem o arquivo ASCII intermediário.

COMO FUNCIONA UM CBBS

O quadro de avisos geralmente é "hospedado" por um microcomputador ligado, por um ou mais modems, a um tronco telefônico. Para baratear os custos operacionais, a maioria dos CBBS atende simultaneamente a um número limitado de usuários — normalmente entre 1 e 5. À primeira vista, isso parece pouco, porém, como veremos adiante, é uma impressão falsa já que o tempo útil de conexão de cada usuário é extremamente curto: o suficiente para copiar (*download*) as notícias ou programas de interesse para seu disco.

O quadro de avisos central nada mais é que a memória do micro que o gerencia. Hoje, a maioria dos CBBS dispõe de discos rígidos de maior capacidade, mas pode-se operá-los só com dois acionadores de disquetes como memória auxiliar. Nesse caso, entretanto, a velocidade e a capacidade do sistema são significativamente diminuídas.

Muitos dos CBBS recentemente surgidos no Brasil constituem iniciativa de voluntários. Seguindo a tradição dos entusiastas da microinformática, eles são gratuitos (embora existam alguns, como o Sampa CBBS, em São Paulo, que se tornaram tão grandes que foi necessário fixar uma pequena taxa). Um dos sérios problemas desse sistema é impedir que pessoas não autorizadas tenham acesso a ele. Usualmente, isso é feito atribuindo-se uma senha a cada membro. Contudo, essa é uma questão difícil de ser solucionada, tanto que nem mesmo o Pentágono norte-americano vem obtendo sucesso no sentido de evitar que pessoas não autorizadas penetrem em seus monstruosos "mainframes". Na verdade, muitos CBBS recebem de braços abertos todos os intrusos.

PADRÕES

Evidentemente, a distância física entre o usuário e o CBBS não representa nenhum obstáculo, já que um computador pode perfeitamente se comunicar com outro, em qualquer parte do mundo, desde que ambos tenham modems compatíveis. Utilizando o equipamento correto, você poderá se beneficiar de mais de 1.500 serviços de troca de mensagens espalhados pelos EUA, Canadá e Europa. Em razão de não existirem padrões comuns de comunicação entre europeus e

norte-americanos, o usuário que desejar contatar CBBS dessas regiões deverá dispor de modems diferentes.

O padrão adotado na Europa é o CCITT, sigla de uma agência da ONU denominada *Comité Consultatif International Téléphonique et Télégraphique*. Essa agência estabelece convenções internacionais para telecomunicações que podem também ser seguidas internamente por um país-membro. Indiretamente, isso acaba facilitando a comunicação entre todos os proprietários de computadores. Cada país, porém, pode ter simultaneamente outras convenções internas, determinadas e aprovadas por órgãos estatais, como o Contel (Conselho de Telecomunicações), no Brasil.

Nos EUA, o chamado padrão Hayes é o mais comum para CBBS privados (seu nome deriva da fábrica dos modems mais populares e baratos do país). O padrão Bell também é muito usado.

VELOCIDADES DE TRANSMISSÃO

As agências de normatização, nacionais e internacionais, deixam aos fabricantes de modems a criação de muitas



outras convenções. Os modems são classificados em função de diversos parâmetros operacionais e de desempenho como, por exemplo, a velocidade de transmissão. A grande maioria dos CBBS — os mais baratos — opera com modems de 300 bauds, isto é, eles recebem e transmitem dados à velocidade de 300 bits por segundo. (O nome da unidade de medida *baud* é uma homenagem ao engenheiro francês Baudot, considerado o inventor de um código para telex.) Isso equivale, aproximadamente, a um bit enviado a cada 3.3 milissegundos. Já os sistemas mais avançados de telecomunicações digitais seguem o padrão 1200/75 baud: o computador central transmite dados à velocidade de 1200 bauds e o usuário os envia a apenas 75 bauds. Sistemas como o Cirandão, da Embratel, e o Videotexto utilizam essa convenção (seu objetivo é reduzir o tempo de ocupação do computador central, devido ao custo elevado, e impor os gastos da interação ao usuário).

As taxas de transmissão normalmente são múltiplos de 300 (300, 600, 1200, 2400 etc.), duplicando progressivamente até atingir 9600 bauds, que é o máximo em uso atualmente.

Em muitos computadores, dependendo dos padrões definidos pelo CCICT para interfaces seriais RS-232, as voltagens de representação dos números binários 0 e 1 são, respectivamente, +12 e −12 V. Os micros que usam as voltagens +5 e 0 V necessitam de um conversor, que geralmente é barato.

BAUDS E BYTES

Cada caractere de informação transmitido por um modem compõe-se, basicamente, de um bit de início (*start bit*), dos bits de dados (*data bits*) de um bit de paridade (*parity bit*), usado em testes de erros de transmissão, e de um bit de término (*stop bit*). Como vemos, existem mais bits do que os oito de um byte; assim, para saber quantos bytes (caracteres de texto) são enviados por segundo, tomando como padrão uma taxa de 300 bauds, por exemplo, dividimos esse número por 11.

Embora não haja uma seqüência "natural" nem uma quantidade preestabelecida para os diferentes bits que formam o caractere de transmissão, é importante que tanto o emissor quanto o receptor reconheçam a convenção seguida. Além disso, ambos devem ser capazes de formatar, enviar ou receber os dados conforme a taxa padronizada.

A velocidade adotada por um CBBS será escolhida levando-se em conta a necessidade de se reduzir os custos do serviço, até porque os *Computerized Bulletin Board Service* não dispõem dos recursos de hardware das grandes empresas de telecomunicações. O fato de serem empregados modems mais baratos implica o uso de uma taxa menor de transmissão a fim de evitar erros causados por interferências elétricas, normalmente filtradas por equipamentos sofisticados. Com um modem barato operando a altas taxas, qualquer pulso na rede poderia ser interpretado como um bit. Em velocidades pequenas, o pulso que define um bit é mais longo e, portanto, mais fácil de identificar.

De qualquer modo, o ruído será sempre um problema difícil de se resolver. Com a entrada em serviço das redes digitais de comunicações baseadas em fibras ópticas, espera-se que os sinais sejam mais "limpos" e rápidos.

DUPLEX

Denominamos *full-duplex* ou *half-duplex* ao parâmetro que varia de modem para modem e diz respeito à simultaneidade das funções de transmissão e recepção. Full-duplex significa que um determinado modem é capaz de receber e transmitir informações ao mesmo tempo; half-duplex indica que ele executa apenas uma função de cada vez.

Evidentemente, um modem half-duplex é mais barato e adequado a trabalhos com micros domésticos, ou até a algumas aplicações profissionais mais simples. Como é preciso aguardar uma transmissão terminar antes de iniciar a recepção (ou vice-versa), os erros cometidos na operação poderão trazer alguns inconvenientes. A possibilidade de se consumir um tempo excessivo praticamente desaconselha o uso deste equipamento para trabalhos profissionais.

Os CBBS se tornaram viáveis graças à sofisticação do hardware e software de comunicações. O maior passo nesse sentido foi a introdução de modems ou recursos de autodiscagem (*auto-dialing*, em inglês) e auto-resposta (*auto-answering*). A partir dessas inovações, as consultas ao computador hospedeiro do CBBS passaram a ser respondidas automaticamente com uma linha conectada entre receptor e transmissor.

SOFTWARE PARA COMUNICAÇÕES

Além do computador, do modem e do telefone, faz-se necessário um software especial que transforme o micro em um terminal de dados.

Classificamos o software de comunicações em: *burros* e *inteligentes*. O inteligente — essencial para quem deseja operar um CBBS — permite descarregar o programa (além de texto de mensagens) e armazená-lo automaticamente no disco. Já o software burro possibilita apenas a comunicação simples, sem qualquer forma de armazenamento ou listagem. Certos softwares inteligentes oferecem recursos extremamente úteis, entre eles o reconhecimento automático — através do programa — do protocolo seguido pelo computador hospedeiro e a armazenagem e discagem automática de vários números de CBBS.

A função básica do programa é a de permitir que o micro transmita e receba sinais (caracteres) pelo modem conectado ao sistema, mantendo o sincronismo entre o teclado, o vídeo e a memória. O software orienta o computador por intermédio de comandos relativos à forma com que os sinais serão transmitidos e recebidos.

Muitas pessoas não conseguem entender como dois computadores aparentemente incompatíveis podem se comunicar. E são essas mesmas pessoas que costumam criticar os fabricantes de micros por constantemente ignorarem os padrões que facilitam a interligação entre equipamentos distintos.

Por meio do CBBS, temos a falsa impressão de que um computador está identificando os comandos enviados por outro diferente (um Apple e um MSX, por exemplo). Na verdade, isso só acontece porque a maioria dos CBBS implementa seu software de acesso por meio de menus relativamente simples.

Com eles, o usuário seleciona uma opção pressionando uma tecla ou digitando uma linha de comando. Do outro lado, um programa especial "varre" o modem de recepção, de modo a captar os sinais enviados e a interpretar corretamente o pedido. A partir daí, os comandos adequados passam a ser acionados pelo computador hospedeiro.

EXPLORANDO OS CBBS

O primeiro passo para explorar todos os recursos dos CBBS à sua disposição é adquirir um modem adequado. Conforme abordamos, os CBBS normalmente adotam um tipo padrão de 300 bauds, com protocolo CCICT ou Hayes. Os sistemas comerciais, como o Videotexto e o Cirandão, utilizam o padrão 1200/75. Assim, o ideal é que o usuário compre um modem que tenha uma chave de seleção para cada modalidade.

O segundo passo consiste em provi-

denciar um software de comunicação (se ele já não tiver sido fornecido juntamente com o modem).

O terceiro passo é descobrir os números de telefone dos CBBS e seus detalhes técnicos de acesso. Esses dados podem ser encontrados em revistas especializadas de microcomputação.

Superadas todas as questões, passaremos à parte prática da operação. Conecte o modem à linha telefônica e ao microcomputador e execute o software de comunicação. Caso a versão utilizada disponha de autodiscagem, todo esse trabalho será feito automaticamente, uma vez especificado o CBBS que se deseja acessar. Do contrário, bastará ligar o número telefônico do mesmo e aguardar o sinal de recepção (geralmente um som agudo de poucos segundos denominado *portadora*). Recebido o aviso, pressione o botão que conecta o modem à linha (se o equipamento for de conexão direta), ou coloque o bocal do telefone nas "orelhas" de acoplamento (se for um modem acústico).

Para termos acesso a alguns CBBS, é necessário chamar o número desejado, deixar tocar uma vez, e, em seguida, discar novamente esse número. Só então o procedimento já descrito poderá ser observado.

Muitos CBBS funcionam de maneira semelhante. No início da operação, o usuário recebe uma mensagem que indica seu contato com o CBBS. Depois, deve enviar uma senha de entrada e/ou o número ou nome de identificação — principalmente se o uso do CBBS implica pagamento através de assinatura mensal ou anual. Tais sistemas, entretanto, chegam a oferecer, por um tempo limitado, algumas informações para convidados e "intrusos", isto é, pessoas ainda não registradas no serviço.

Os CBBS gratuitos, dependendo do tipo, podem solicitar seu nome e cidade ou endereço e número telefônico, bem como algumas especificações do micro usado (modelo, tamanho da tela etc.). Esses detalhes são importantes para que o computador hospedeiro saiba como operar durante a comunicação.

Na fase seguinte, você receberá informações do CBBS: horários de funcionamento, limite de tempo para cada chamada etc. Tais dados interessam, sobretudo, aos CBBS voluntários, já que algumas chamadas são feitas em horários inconvenientes (desagradando o operador, que geralmente utiliza o telefone de sua própria casa para isso).

Assim como o videotexto, a maioria dos CBBS é operada através de menus. A escolha de uma opção em um menu poderá levar a menus secundários e terciários etc. ou à execução da opção. Muitos CBBS possuem uma seção que permite aos novos usuários registrar seus dados (endereço, nome, telefone e senha de acesso). Outras opções oferecidas podem incluir uma seção técnica, comunicações com o gerente etc.

O elemento fundamental do CBBS está no quadro que contém avisos e notas para conhecimento geral. Isso, porém, não exclui a possibilidade de existir também uma seção de mensagens pessoais: "caixas postais", com acesso limitado por senhas secretas. Esse sistema permite deixar informações reservadas para outros usuários ou formar subquadros de avisos dentro do CBBS.

TERMINOLOGIA

Há vários termos técnicos que, embora comuns no campo da comunicação entre computadores, são pouco conhecidos em outros ramos da informática. Devido a isso, às vezes, cria-se uma certa confusão sobre seu significado.

A seguir, uma lista dos termos que normalmente são mal interpretados:

Videotexto - Sistema especial de comunicação entre computadores, oferecido como um serviço público pelas companhias telefônicas. Através de uma linha comum, o computador central envia ao usuário a informação (organizada em páginas ou telas). Esta é recebida por um terminal especial de videotexto ou um micro qualquer, que podem responder fazendo novas solicitações de dados. No Brasil, tem-se acesso aos sistemas de videotexto por meio de uma assinatura mensal junto à companhia telefônica de cada cidade. Na Europa, esse serviço, implantado em vários países, é denominado *Prestel*.

Teletexto - Sistema semelhante e mais barato que o videotexto. Aqui, entretanto, a informação é enviada apenas em um sentido (do computador central para o terminal), aproveitando o intervalo disponível entre dois quadros exibidos pela TV. Este sistema não existe no Brasil. Exemplos no exterior são os CEEFAX e o ORACLE, utilizados na Inglaterra. Aparelhos de TV de algumas marcas já são fabricados com a interface embutida.

Videotex - Termo geral que engloba o videotexto, o teletexto e os CBBS. Criou-se uma certa confusão depois que alguns CBBS evoluíram e se transformaram em uma *rede*, apesar de conservarem o mesmo nome.

Rede - Em sua forma mais restrita, refere-se à ligação direta entre vários computadores com objetivos específicos (normalmente para fins comerciais ou de pesquisa, como a rede brasileira RENPAC). As redes dividem-se em dois tipos: a local (LAN, ou *Local Area Network*), que, geralmente, envolve apenas a ligação direta entre micros e seus periféricos ou a um computador central, e a telemática (WAN, ou *Wide Area Network*), que inclui ligações remotas, via rede telefônica, microondas ou satélites. Através da LAN, os usuários podem compartilhar recursos como discos e impressoras. A WAN, por sua vez, permite a intercomunicação e a consequente troca de mensagens e transações (como em uma rede de automação bancária, que liga várias agências aos computadores centrais).

Uma rede pode ser privada ou pública. Existem vários exemplos de redes públicas, como o Cirandão, da Embratel, ou o CompuServe, nos EUA, com mais de 250.000 usuários.

Nos últimos anos, o termo rede vem perdendo sua especificidade, passando a significar qualquer forma de intercomunicação realizada entre computadores. Por esse conceito, todos os serviços do tipo videotex também poderiam ser considerados uma rede.

Telex e teletex - Serviços em fase de integração a redes de computadores e que, por isso, ainda são confundidos. O telex é um sistema telefônico de comunicação de textos, de baixa velocidade (15 bauds), gerenciado por uma central especial computadorizada. Existem interfaces que permitem dar a um micro a função de um terminal de telex (daí serem chamadas de microtelex).

O teletex é uma evolução técnica do telex e, muito provavelmente, seu substituto direto. Utiliza o mesmo sistema de comunicação, porém, a uma velocidade bem maior, de modo a facilitar o uso de micros como terminais. Ainda não foi implantado no Brasil.

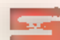
Correio eletrônico - Serviço oferecido por diversos tipos de rede, videotexto ou CBBS. Consiste, basicamente, de "caixas postais" (espaço reservado no disco central a cada assinante), capazes de receber mensagens individuais ou circulares, enviadas por outros usuários. A comunicação é feita a nível privado, e protegida por senhas. Em alguns países, o sistema é prerrogativa do serviço estatal de correios. No Brasil, como em outras nações, esse monopólio já foi rompido (Cirandão, Mensagem, Videotexto).

UM EDITOR MUSICAL (2)

Continue digitando nosso programa editor de melodias. Com as listagens fornecidas no próximo artigo da série, ele ficará completo e você poderá dar vazão a todo o seu talento musical.

Apresentamos aqui a segunda parte de nosso editor musical. Carregue a primeira parte da fita cassete ou disco e digite, na sequência, a listagem dada abaixo. Depois, grave as duas partes para juntá-las à que iremos publicar no próximo e último artigo da série — onde você encontrará, também, instruções detalhadas para uma eficiente utilização do programa.

Como você terá oportunidade de observar, à medida que for digitando, o programa é composto por várias sub-rotinas chamadas do menu principal. É possível ter uma idéia da função de cada uma delas lendo os itens do menu — que pode ser exibido na tela ainda que o programa esteja incompleto.



```
2000 CLS
2010 GOSUB 2500
2140 PRINT 'ct;" Notas tecladas
 (";maxnotes;"Max.)"''
2160 INPUT "Entre Notas - <RET>
 para acabar";N$
2175 IF LEN (N$)=0 THEN  GOTO 1
90
2180 FOR i=1 TO LEN (N$): IF (N
$(i)<"0" OR N$(i)>"9") AND (N$(
i)<>".") THEN  GOTO 2000
2190 LET N=VAL (N$)
2200 IF INT (N/1000)>11 AND INT
 (N/1000)<>-4 THEN  GOTO 2000
2210 IF N<0 THEN  GOTO 2280
2220 LET M=INT (N/100): LET D=N
-M*100
2230 LET O=M-INT (M/10)*10: IF
O<1 OR O>7 THEN  GOTO 2000
2240 LET M=INT (M/10)+(O-1)*12-
36
2260 LET ct=ct+1: LET t(2*ct-1)
=D: LET t(2*ct)=M
2270 GOTO 2000
2280 LET M=-4: LET D=0-(N-M*100
)
2290 IF M<>-4 THEN  GOTO 2000
2310 GOTO 2260
2500 PRINT "Entre a Nota na for
ma <Numero>, <Oitava>, <Duracao
>."'
2520 PRINT "C - 0   E - 4   G#-
 8"'"C#- 1   F - 5   A - 9"'"D
- 2   F#- 6   A#- 10"'"D#- 3
G - 7   B - 11"
2560 PRINT '"PAUSA e -4"''"Semi
colcheia=1   Minima   = 8"'"Col
cheia    =2   Semibreve=16"'"Se
minima    =4   Oitava   =1 a 7"
2600 PRINT '"Duracao DEVE ter 2
 digitos"'"ex. 3304-D#, Colchei
a 3a. oitava"'"ex. 6408-F#, Min
ima 4a. oitava"
2630 RETURN
3000 CLS
3010 PRINT "Tocar musica"
3020 PRINT : PRINT : PRINT
3030 PRINT "Digite Tempo -  (1-
15)   ";
3040 INPUT S
3050 IF S<1 OR S>15 THEN  GOTO
3000
3060 LET tempo=0.02*(16-S)
3070 FOR i=1 TO ct
3080 LET D=t(2*i-1): LET M=t(2*
i)
3090 IF m=-4 THEN  GOTO 3120
3100 SOUND D*tempo,M
3110 GOTO 3130
3120 PAUSE 50*D*tempo
3130 NEXT i
3140 RETURN
4000 CLS
4010 PRINT "EDITOR MUSICAL"''''
"D - Mostrar todas as notas"''"
E - Editar uma nota"''"I - Inse
rir uma nota"''"X - Apagar uma
nota"''''"R - Retornar ao menu
principal"
4090 PRINT ''''"Escolha uma opca
o - ";
4100 LET O$=INKEY$: IF O$="" TH
EN  GOTO 4100
4105 IF CODE (O$)<97 THEN  LET
O$=CHR$ (CODE (O$)+32)
4110 IF O$<>"d" AND O$<>"e" AND
 O$<>"i" AND O$<>"r" AND O$<>"x
" THEN  GOTO 4000
4120 IF O$="r" THEN  RETURN
4130 IF O$="e" THEN  GOTO 4300
4134 IF O$="i" THEN  GOTO 4700
4136 IF O$="x" THEN  GOTO 4800
4140 CLS
4150 FOR i=1 TO ct
4160 LET M=t(2*i): LET D=t(2*i-
1)
4170 LET O=INT ((M+36)/12)+1
4180 LET N=(M+36)-(O-1)*12
4190 PRINT i;" Nota- ";N;" Oit.
- ";O;" Dur.- ";D
4195 POKE 23692,255
4200 IF i=20*INT (i/20) THEN  G
OTO 4220
4210 GOTO 4250
4220 PRINT '"Qualquer tecla par
a continuar ";
4230 PAUSE 0
4240 PRINT
4250 NEXT i
4260 PRINT '"Qualquer tecla par
a retornar ";
4270 PAUSE 0
4280 GOTO 4000
4320 CLS
4330 GOSUB 2500
```



```
650 P=P-36
660 IF P=1ANDC<6THEN C=C+1:OC$=
MID$(STR$(C),2)
670 IF P=2ANDC>1THEN C=C-1:OC$=
MID$(STR$(C),2)
680 IF(P=3ORP=6)AND LE>1 THEN L
E=LE-1:LE$=MID$(R1$,LE,1)+" "
690 IF(P=4ORP=7)AND LE<7THEN LE
```



- SEGUNDA PARTE DO PROGRAMA EDITOR
- GRAVAÇÃO E LEITURA
- DIGITAÇÃO DAS SUB-ROTINAS

```
=LE+1:LE$=MID$(R1$,LE,1)+" "
700 IF P=5 THEN I$="p":P$="":GO
TO 750
710 IF P=6ORP=7 THEN LE$=LEFT$(
LE$,1)+"."
720 IF P=8 THEN RETURN
730 IF P=9 AND NN>0 THEN NN=NN-
1
740 GOTO 510
750 U$="L":IF I$>="a" AND I$<="
g" THEN P$=CHR$(ASC(I$)-32) ELS
E IF I$<>"p"THEN P$=I$+"#"ELSE
P$="":U$="R"
760 PL$="V15T"+STR$(TE)+"O"+OC$
+U$+L2$(INSTR(R1$,LEFT$(LE$,1))
)
770 IF MID$(LE$,2,1)="."THEN P$
=P$+"."
780 PL$=PL$+P$:PLAY PL$
790 NN=NN+1:N$(NN)=I$+OC$+LE$
800 GOTO 580
810 CLS:COLOR 4,15
820 LOCATE 3,2:PRINT"ENTRADA MA
NUAL DE NOTAS";
830 LOCATE 3,18:PRINT"'m'  RETO
RNAR-MENU PRINCIPAL"
840 LOCATE 3,19:PRINT"USE FORMA
TO:alh.('.'opcional)
850 GOSUB 360:LOCATE 3,21:PRINT
"ENTRE STRING NOTA:";
860 IF NN=MX THEN LOCATE 1,23:P
RINT"MAXIMO NUMERO DE NOTAS COL
OCADO!":FOR S=1TO1000:NEXT:RETU
RN
870 LINE INPUT A$
880 IF A$="" THEN 870
890 IF A$="m"OR A$="M" THEN RET
URN
900 IF LEN(A$)>4ORLEN(A$)<3 THE
N 970
910 IF (INSTR(R3$,LEFT$(A$,1)))
=0 THEN970
920 IF LEFT$(A$,1)="p" THEN A$=
"p"+" "+MID$(A$,3):GOTO 940
930 IF (INSTR(R2$,MID$(A$,2,1))
)=0 THEN 970
940 IF (INSTR(R1$,MID$(A$,3,1))
)=0 THEN 970
950 IF MID$(A$,4,1)<>"." THEN A
$=LEFT$(A$,3)+" "
960 GOTO 990
970 LOCATE 21,21:PRINT"ENTRADA
ILEGAL!";CHR$(7)
980 LOCATE 21,21:PRINT"
        ":GOTO 850
990 NN=NN+1:N$(NN)=A$
1000 GOTO 850
1010 IF NN=0 THEN RETURN
1020 COLOR 1,15:LOCATE 0,0:PRIN
T "MODO EDIÇÃO- 1/2/3/4
     "
1030 LOCATE 0,1:PRINT STRING$(4
0,32)
1040 LOCATE 0,2:PRINT STRING$(4
0,32)
1050 LOCATE 0,18:PRINT STRING$(
40,32)
1060 LOCATE 0,3:PRINT STRING$(4
0,32)
1070 LOCATE 2,20:PRINT"1:APAGAR
 NOTAS";
1080 LOCATE 2,21:PRINT"2:INSERI
R NOTAS";
1090 LOCATE 2,22:PRINT"3:ALTERA
R NOTAS";
1100 LOCATE 2,23:PRINT"4:CONTIN
UA";
1110 A$=INKEY$:IF A$<"1"ORA$>"4
"THEN 1110
1120 OP=VAL(A$)
1130 ON OP GOTO 1150,1250,1410,
1140
1140 RETURN
1150 LOCATE 12,0:PRINT"1-APAGAR
 NOTAS"
1160 LOCATE 0,2:INPUT"INICIA NA
 NOTA";ST
1170 IF ST=0 THEN 1010
1180 IF ST>NN THEN 1150
1190 LOCATE 0,3:INPUT"QUANTAS A
PAGA (ENTER=1)";ND
1200 IF ND<=0 THEN ND=1
1210 IF ST+ND-1>NN THEN ND=NN-S
T+1
1220 FOR K=1TOND
1230 FOR J=ST TO NN-1:N$(J)=N$(
J+1):NEXT
1240 NN=NN-1:NEXT K:I=SL-1:GOTO
```



```
1010
1250 LOCATE 12,0:PRINT"2 INSERIR NOTAS"
1260 LOCATE 0,2:INPUT"INICIA INSERÇAO APÓS QUE NOTA";ST
1270 IF ST<0 THEN 1010
1280 IF ST>NN THEN ST=NN
1290 LOCATE 0,3:PRINT"TECLE'M'PARA ENCERRAR:";
1300 IF NN=MX THEN LOCATE 2,18:PRINT"MAXIMO NUMERO DE NOTAS COLOCADAS!":FOR D=1TO 2000:NEXT:I=SL 1 GOTO 1010
1310 LINE INPUT N$:IF N$="M"OR N$="m"THEN I=SL-1:GOTO 1010
1320 IF LEN(N$)<3 THEN LOCATE 22,3:PRINT"ILEGAL":GOTO 1290
1330 IF INSTR(R3$,LEFT$(N$,1))=0ORINSTR(R2$,MID$(N$,2,1))=0ORINSTR(R1$,MID$(N$,3,1))=0 THEN LOCATE 22,3 PRINT"ILEGAL":GOTO 1290 NOTAS
1340 IF MID$(N$,4,1)="."THEN N$=LEFT$(N$,4)ELSE N$=LEFT$(N$,3)+" "
1350 IF ST=NN THEN 1380
1360 FOR K=NN+1 TO ST+2 STEP -1
1370 N$(K)=N$(K 1):NEXT
1380 N$(ST+1) N$
1390 ST=ST+1:NN=NN+1
1400 GOTO 1290
1410 LOCATE 13,0:INPUT"3:ALTERAR NOTA",ST
1420 IF ST=0 THEN 1010
1430 IF ST>NN THEN ST=NN
1440 LOCATE 11,2:PRINT N$(ST)
1450 RT=255:LP=-2:I=ST:GOSUB 1620
1460 LOCATE 0,2:PRINT"NOVO VALOR:";:LINE INPUT N$
1470 IF LEN(N$)<3 THEN 1460
1480 IF INSTR(R3$,LEFT$(N$,1))=0ORINSTR(R2$,MID$(N$,2,1))=0ORINSTR(R1$,MID$(N$,3,1))=0 THEN 1460
1490 IF MID$(N$,4,1)=" "THEN N$ LEFT$(N$,4)ELSE N$=LEFT$(N$,3)+" "
1500 N$(ST)=N$:GOSUB 1620 :FOR K=1TO2000:NEXT:RT=0:I=SL-1:LP=0 GOTO 1010
1510 IF NN=0 THEN RETURN ELSE CLS COLOR1,15:LOCATE 1,1:PRINT"LISTAR NOTAS"
1520 C=1:SL=1:EL=NN:P$="N":RT=0
1530 LOCATE 4,3:INPUT"LISTAR NA IMPRESSORA(S/N)",P$
1540 IF P$="S" THEN C=2
1550 LOCATE 4,5:INPUT"INICIO EM (ENTER=  1)";SL
1560 IF SL<=0 THEN SL=NN:EL=NN
1570 LOCATE 4,6:INPUT"FINAL  EM (ENTER=FIM)",EL
1580 IF EL=0 OR EL>NN THEN EL=NN
1590 IF SL>EL THEN SL=EL
1600 CLS:LP=0
```



```
600 P=P-36
610 IF P=1ANDC<3THEN C=C+1:OC$=MID$(STR$(C),2)
620 IF P=2ANDC>1THEN C=C-1:OC$=MID$(STR$(C),2)
630 IF (P=3ORP=6)AND LE>1 THEN LE=LE-1:LE$=MID$(R1$,LE,1)+" "
640 IF (P=4ORP=7)AND LE<5 THEN LE=LE+1:LE$=MID$(R1$,LE,1)+" "
650 IF P=5 THEN I$="p":P$="":GOTO 700
660 IF P=6ORP=7 THEN LE$=LEFT$(LE$,1)+"."
670 IF P=8 THEN RETURN
680 IF P=9 AND NN>0 THEN NN=NN-1
690 GOTO 490
700 IF I$>="a" AND I$<="g" THEN P$=CHR$(ASC(I$)-32) ELSEIF I$<>"p" THEN P$=I$+"#"ELSE P$=""
710 PL$="V31;T"+STR$(TE)+"L"+L2$(INSTR(R1$,LEFT$(LE$,1)))
720 IF MID$(LE$,2,1)="." THEN PL$=PL$+"."
730 PL$=PL$+"O"+OC$+P$:PLAY PL$
740 NN=NN+1:N$(NN)=I$+OC$+LE$
750 GOTO 530
760 CLS
770 PRINT@4,"ENTRADA MANUAL DE NNTAS":PRINT@33,"ENTRE'm'PARA RETORNAR AO MENU"
780 PRINT"USE FORMATO:'a1h.'('.'OPCIONAL)"
790 GOSUB 280:PRINT@416,"ENTRE STRING NOTA."
800 IF NN=MX THEN PRINT@448,"MAXIMO NUMERO DE NOTAS COLOCADO!":FORD=1TO1000:NEXT:RETURN
810 POKE 282,0 PRINT@448:PRINT@462,;:LINE INPUT A$
820 IF A$="" THEN 810
830 IF A$="m"OR A$="M"THEN RETURN
840 IF LEN(A$)>4OR LEN(A$)<3 THEN 910
850 IF (INSTR(R3$,LEFT$(A$,1)))=0 THEN 910
860 IF LEFT$(A$,1)="p" THEN A$="p"+" "+MID$(A$,3):GOTO 880
870 IF (INSTR(R2$,MID$(A$,2,1)))=0 THEN 910
880 IF (INSTR(R1$,MID$(A$,3,1)))=0 THEN 910
890 IF MID$(A$,4,1)<>"." THEN A$=LEFT$(A$,3)+" "
900 GOTO 920
910 PRINT@448,LEFT$(A$,4);" -ENTRADA ILEGAL!":SOUND 1,5:GOTO 810
920 NN=NN+1:N$(NN)=A$
930 GOTO 790
940 IF NN=0 THEN RETURN ELSE POKE 282,0
950 PRINT@448,"1:APAGAR NOTA ","2:INSERIR NOTAS","3:ALTERAR NOTA ","4:CONTINUAR";
960 A$=INKEY$:IF A$<"1"OR A$>"4"THEN 960
970 OP=VAL(A$)
980 ON OP GOTO 1000,1100,1260,990
990 RETURN
1000 IF NN=0 THEN 940 ELSE CLS:INPUT"INICIA NA NOTA";ST
1010 IF ST=0 THEN 940
1020 IF ST>NN THEN 1000
1030 PRINT@64,"QUANTAS APAGA(ENTER=1)";:INPUT ND
1040 IF ND<=0 THEN ND=1
1050 IF ST+ND-1>NN THEN ND=NN-ST+1
1060 FOR I=1 TO ND
1070 FOR J=ST TO NN 1:N$(J)=N$(J+1):NEXT
1080 NN=NN-1
1090 NEXT I:RETURN
1100 CLS:PRINT@7,"MODO INSERCAO DE NOTAS"
1110 PRINT@64,"INICIA INSERCAO APOS QUE NOTA":INPUT ST
1120 IF ST<0 THEN 940
1130 IF ST>NN THEN ST=NN
1140 PRINT@64,"TECLE'm'QUANDO VOCE TERMINAR":PRINT
1150 IF NN=MX THEN PRINT"MAXIMO NUMERO DE NOTAS COLOCADO!":FOR D=1 TO 1000:NEXT:RETURN
1160 INPUT N$:IF N$="M"OR N$="m"THEN RETURN
1170 IF LEN(N$)<3 THEN PRINT"ILEGAL":GOTO 1160
1180 IF INSTR(R3$,LEFT$(N$,1))=0ORINSTR(R2$,MID$(N$,2,1))=0ORINSTR(R1$,MID$(N$,3,1))=0THEN PRINT"ILEGAL":GOTO 1160
1190 IF MID$(N$,4,1)="."THEN N$=LEFT$(N$,4) ELSE N$=LEFT$(N$,3)+" "
1200 IF  ST=NN THEN 1230
1210 FOR I=NN+1 TO ST+2 STEP-1
1220 N$(I)=N$(I-1):NEXT
1230 N$(ST+1)=N$
1240 ST=ST+1:NN=NN+1
1250 GOTO 1150
1260 CLS:PRINT"MUDAR QUE NOTA";:INPUT ST
1270 IF ST=0 THEN 940
1280 IF ST>NN THEN ST=NN
1290 PRINT:PRINT"VALOR ATUAL:":A$=N$(ST):RT=255:GOSUB 350
1300 PRINT N$(ST)
1310 PRINT@320:PRINT@320,"";:INPUT"NOVO VALOR:";N$
1320 IF LEN(N$)<3 THEN 1310
1330 IF INSTR(R3$,LEFT$(N$,1))=0ORINSTR(R2$,MID$(N$,2,1))=0ORINSTR(R1$,MID$(N$,3,1))=0THEN 1310
1340 IF MID$(N$,4,1)="." THEN N$=LEFT$(N$,4)ELSE N$=LEFT$(N$,3)+" "
1350 N$(ST)=N$:FORK=1TO2000:NEXT:GOTO940
1360 IF NN=0THEN RETURN ELSE CLS PRINT@7,"OPCAO LISTAR NOTAS"C=0
1370 IF(PEEK(65314)AND1)=1 THEN 1400
1380 POKE 282,255:PRINT@64,"",INPUT"LISTAR NA IMPRESSORA(S/N)";P$
1390 IF P$="S" THEN C=2 POKE 150,1
1400 PRINT@128,"INICIO NA NOTA";:INPUT ST
1410 IF ST<=0 THEN ST=NN
1420 IF ST>NN THEN ST=NN
1430 CLS:LP=0
```



OS SEGREDOS DO TRS-80 (5)

Aprenda a explorar todas as possibilidades do teclado do TRS-80 e a utilizá-lo de forma criativa. Comandos PEEK e POKE garantirão o sucesso de suas experiências.

Nos artigos anteriores desta série, examinamos os recursos "secretos" do micro TRS-80 para a saída de vídeo. Entre outras coisas, vimos como gerar pseudo-sprites e como levar cópias de telas para a memória e vice-versa.

O teclado do TRS-80 também tem várias características interessantes, que não estão bem documentadas nos manuais de operação ou mesmo nos livros mais elementares sobre o assunto. Neste artigo, você aprenderá alguns truques que podem ser executados com facilidade, como o bloqueio da tecla <**BREAK**>, a implementação de um cursor piscante, a programação da repetição automática de teclas e a entrada direta de símbolos gráficos pelo teclado.

Esses recursos são possíveis porque a memória do TRS-80 contém o mapa do teclado, assim como o do vídeo. O teclado desse micro é do tipo matricial, ou seja, ao se pressionar uma tecla, ela aciona o cruzamento entre dois condutores: um colocado nas colunas, e outro, nas linhas. Isso gera um código binário, que é recuperado pelo software processador do teclado. Este efetua uma varredura a intervalos de alguns milissegundos, passando por todos os fios horizontais e verticais. O número obtido é depositado em uma locação específica da memória de teclado.

Diversas outras locações da memória RAM reservada para a área de trabalho do interpretador BASIC são dedicadas ao teclado. Através de comandos **PEEK** e **POKE**, é possível verificar, e mesmo alterar, seu conteúdo, produzindo curiosos e variados efeitos.

BLOQUEIO DA TECLA <BREAK>

Como você já sabe, a tecla <**BREAK**> do TRS-80 interrompe um programa BASIC em execução, independentemente do que ele estiver fazendo, e retorna o controle ao interpretador.

A facilidade de interromper o programa pode ser um inconveniente quando se deseja torná-lo imune a erros de operação ou se quer impedir que alguém o liste. Imagine a seguinte situação: você colocou um programa educativo nas mãos de uma criança; ao aparecer a mensagem "PRESSIONE QUALQUER TECLA PARA CONTINUAR", ela aciona justamente a tecla <**BREAK**>! A culpa, com certeza, é sua...

Para evitar ocorrências desse tipo, convém bloquear o efeito da tecla <**BREAK**> através de comandos **POKE** nas locações de memória 16396 e 16397:

```
POKE 16396,175:POKE 16397,201
```

Para ativar novamente a tecla, basta utilizar o comando:

```
POKE 16396,201
```

Esse truque funciona em qualquer micro compatível com os modelos I e III da linha TRS-80, e pode ser empregado tanto como comando direto quanto dentro de um programa.

Há um problema, entretanto. Se você estiver usando o BASIC de disco, e tentar acessá-lo durante a desativação da tecla <**BREAK**>, o computador poderá "congelar". Para evitar que isso aconteça, aconselha-se recorrer a outra locação de memória para dar o comando **POKE**. Não se esqueça, porém, de que esta varia de acordo com o sistema operacional de disco (DOS) empregado:

DOS	DESATIVA	ATIVA
TRSDOS 2.3	POKE 23886,0	POKE 23886,1
NEWDOS 2.1	POKE 23461,0	POKE 23461,1
NEWDOS 80	POKE 19408,0	POKE 19408,1

A maioria dos micros nacionais da linha TRS-80 utiliza sistemas compatíveis com o TRSDOS (pronuncia-se *trisdós*). Se seu micro usa a versão CP/M, você não poderá desativar a tecla <**BREAK**> por meio da operação sugerida.

Um aviso final: se seu programa ainda não foi gravado, tenha muito cuidado ao usar esse truque, assegurando que a tecla <**BREAK**> possa ser reativada. Caso contrário, você poderá ficar na desagradável situação de nunca mais poder listar seu próprio programa!

AUTO-REPETIÇÃO

Os modelos da linha TRS-80 I e III (como o Prológica CP-500) não têm a capacidade de repetição automática das teclas — ou seja, as teclas não se auto-repetem durante o período em que estão sendo pressionadas.

Este é um recurso útil para uma série de aplicações: por exemplo, jogos em que o movimento de um cursor ou o disparo de uma metralhadora são controlados por teclas auto-repetitivas.

Há uma maneira, porém, de saber se uma tecla continua sendo pressionada ou não, o que nos permite implementar uma rotina de repetição. Se o valor dado por um **PEEK(14591)** for igual a 0, nenhuma tecla está sendo pressionada; se esse valor for maior que 0, uma tecla está sendo pressionada.

Para entender como funcionaria um laço simples de repetição, digite:

```
10 PRINT PEEK(14591);:GOTO 10
```

Ao ser executado, o programa imprime uma seqüência de zeros na tela. Quando uma tecla é pressionada, esse valor muda. Note que cada tecla ou combinação de teclas (pressionadas junto) tem um valor diferente, que não corresponde ao valor ASCII da tecla.

Para aplicar esse expediente, identifique os números das teclas que deseja usar para direcionar a lógica de determinado programa e introduza vários **IF** dentro do mesmo. Se você quiser utilizar os valores ASCII gerados por cada tecla pressionada, precisará de um **IF** para testar se a tecla continua pressionada, e outro para localizar essa tecla.

Como exemplo, execute este programa. Ele desloca um cursor pela tela, sob o controle de quatro teclas (flechas).

```
10 CLS:X=64:Y=24
15 SET(X,Y)
20 IF T%>0 AND PEEK(14591)>0
THEN 50
25 T$=INKEY$:IF T$="" THEN 25
30 T%=ASC(T$)
50 IF T%=8 THEN X=X-1:GOTO 15
60 IF T%=9 THEN X=X+1:GOTO 15
70 IF T%=91 THEN Y=Y-1 GOTO 15
80 IF T%=10 THEN Y=Y+1:GOTO 15
90 GOTO 25
```

A linha 10 do programa define a posição inicial do cursor (um pequeno ponto na tela), "aceso" pelo comando **SET**, com as coordenadas X e Y na tela.

A linha 20 verifica se alguma tecla está sendo pressionada — ou seja, se o valor **T%** (código ASCII da tecla, deter-

- TRUQUES COM O TECLADO
- COMO DESATIVAR A TECLA <BREAK>
- ROTINA PARA A REPETIÇÃO AUTOMÁTICA DAS TECLAS
- CURSOR DE TEXTO PISCANTE
- ENTRADA DIRETA DE SÍMBOLOS GRÁFICOS

minado na linha 30) e o conteúdo da memória 14591 são, simultaneamente, maiores que 0. Em caso afirmativo, o programa pula para a linha 50, que inicia vários testes para identificar qual das teclas com flecha foi pressionada. As coordenadas X e Y são alteradas para mais ou para menos, e um outro ponto é aceso na tela na nova posição. A linha 20 testa mais uma vez se essa tecla continua pressionada, executa um deslocamento e assim por diante.

Se nenhuma tecla estiver sendo pressionada, o programa vai para a linha 25, que cria um laço de varredura do teclado com a função **INKEY$**. Essa linha se repete até que se pressione alguma tecla. Então, o valor ASCII da tecla é identificado e armazenado em **T%**, pela linha 30. A partir daí, o micro se comporta como se tivesse a capacidade de repetição automática das teclas.

Observe que, se nenhuma das quatro teclas com flecha tiver sido pressionada, o programa retorna ao laço de espera situado na linha 25.

Ao executar esse programa, tenha o cuidado de não ultrapassar os limites da tela com o cursor, para que não ocorra um erro na linha 15. Se quiser melhorar o programa, introduza testes para as alterações de X e Y, de modo a impedir que eles ultrapassem os valores mínimo e máximo da tela.

O CURSOR PISCANTE

Normalmente, o cursor de texto do TRS-80 é um traço de sublinhar, não piscante. Não é fácil localizá-lo com a tela cheia de texto. A dificuldade aumenta em aplicações de deslocamento bidimensional do cursor, sobretudo se ele se sobrepõe a um trecho gráfico.

Um bom recurso consiste em fazer o cursor piscar repetidamente, distinguindo-se dos caracteres de fundo. No TRS-80, porém, só se obtém esse efeito por meio de software, já que nenhum de seus comandos permite alterar a função do cursor. Portanto, a rotina precisa ser suficientemente rápida para não "segurar" o usuário que está digitando o texto a uma certa velocidade.

Um problema adicional é fazer o cursor piscar quando está sobre um caractere já impresso. Nesse caso, devemos copiar o valor do caractere em uma variável, antes de piscar, e recolocá-lo na tela. Os comandos **PEEK** e **POKE** desta rotina encarregam-se disso:

```
100 T$=INKEY$  IF T$<>"" THEN
120
110 POKE X%,95 POKE X%,C%:GOTO
100
120 RETURN
```

Antes de chamar a rotina, coloque a posição atual do cursor em **X%**. Este é um valor entre 15360 e 16383, números que correspondem às locações da memória de vídeo na RAM. Para obtê-lo, examina-se o conteúdo das memórias 16416 e 16417. O valor ASCII do caractere nessa posição deve ser colocado em **C%**. Assim, a rotina poderá fazer o cursor piscar, após verificar o código armazenado na locação **X%** de vídeo:

```
10 X%=PEEK(16417)*256+PEEK
(16416)
20 C%=PEEK(X%)
30 GOSUB 100 PRINT T$; GOTO 10
```

A linha 30 chama a rotina e imprime o caractere que se pressionou, retornado pela sub-rotina em **T$**. A volta à linha 10 tem o efeito de deslocar o cursor uma posição à direita no vídeo, reiniciando a rotina de piscar o cursor.

Essa rotina permite ainda que se modifique a forma do cursor. Note que usamos o código ASCII 95, que corresponde ao traço de sublinhar. Para colocar na tela um retângulo sólido, substitua 95 por 191, na linha 110.

GRÁFICOS PELO TECLADO

Como vimos em artigo anterior (página 1259), o emprego da rotina **INKEY$** no lugar do comando **INPUT** possibilita a entrada dos caracteres gráficos do TRS-80 diretamente pelo teclado. Assim, se usarmos uma combinação das teclas **<SHIFT>** e **<LINEFEED>** (tecla para baixo), teremos o mesmo efeito da tecla **<CONTROL>** de outros computadores, e poderemos gerar códigos ASCII diferentes, pelo teclado. Se pressionarmos, por exemplo, a tecla A, juntamente com as anteriores, geramos o código ASCII 1. Detectando esse código na rotina **INKEY$**, bastará somá-lo ao valor 128 para obtermos em **T$** (variável de saída) o gráfico correspondente, que também será impresso na tela.

Substitua a linha 130 por:

```
130 IF ASC(T$)<32 THEN T$=CHR$(
T%+128)
140 RETURN
```

MICRO DICAS

VARREDURA DO TECLADO

A função **INKEY$** é bastante útil para se detectar, dentro de um programa, se alguma tecla foi pressionada. Mas, às vezes, ela não é suficiente para satisfazer certas necessidades do programador — como indicar se duas ou mais teclas foram pressionadas simultaneamente, ou detectar o acionamento de uma tecla que não gera um código através do **INKEY$**, como **<SHIFT>**.

Conhecendo a organização do teclado, porém, esse tipo de verificação não oferece dificuldade. O teclado é disposto na forma de uma matriz. Quando uma tecla é pressionada, um determinado valor numérico entre 1 e 128 (potências de 2) é depositado na memória RAM correspondente a uma fileira (um dos seguintes endereços: 14337, 14338, 14340, 14344, 14352, 14368, 14400 ou 14464). A correspondência entre tecla e endereço nos permite identificar a tecla ou teclas pressionadas. Por exemplo, a tecla **<SHIFT>**, quando pressionada, gera o código 1 na locação de memória 14464.

Quando duas ou mais teclas são pressionadas simultaneamente, gerando códigos no mesmo endereço, o número resultante é a soma dos códigos individuais. Assim, se J, K e L forem pressionadas, o número 4 + 8 + 16 = 28 será colocado no endereço 14338.

Para montar sua própria tabela de correspondência entre teclas e valores, faça um programa de uma linha, usando um **PEEK** para examinar o conteúdo de cada uma das memórias dentro de um laço infinito.

COMPRESSÃO DE TEXTOS (2)

Neste artigo, examinaremos novas técnicas de compressão de textos. Com as várias alternativas dadas, não será difícil reduzir o espaço de memória ocupado por seu jogo.

Como vimos no artigo da página 1332, podemos conseguir um alto índice de compressão de textos através da utilização de um conjunto reduzido de caracteres (só letras maiúsculas e alguns sinais de pontuação) e da compactação dos códigos resultantes em um número menor de bits. Examinamos também o uso da estatística de ocorrência de letras e outros símbolos em um texto, para obter um código progressivo de quatro bits. O procedimento garante uma grande eficiência de compressão (cerca de 45%), envolvendo um programa relativamente simples em BASIC, para codificação e decodificação.

Neste artigo, apresentaremos outros métodos de compressão de textos. Um deles, conhecido como *método chinês*, mostra-se particularmente útil para a compressão de textos muito longos e com repetições freqüentes de um conjunto reduzido de palavras — o que, certamente, não é próprio de programas de aventuras. De qualquer maneira, esse método completa o leque de alternativas de compressão de textos e, dada a sua eficácia, poderá ser aproveitado em outros tipos de programa.

UM SUPERCOMPRESSOR

O algoritmo estatístico de compressão de textos, que estudamos no artigo anterior, toma como base a diferença de frequência das letras em um texto para construir um sistema de códigos em que os caracteres de maior uso têm um número menor de bits. Para facilitar a programação em BASIC desse algoritmo, limitamos a quatro o número mínimo de bits por caractere. Poderíamos, entretanto, utilizar um número ainda menor de bits para as letras mais freqüentes: dois ou três, por exemplo. Com o esquema adotado anteriormente — em que se usa um nibble 0 para assinalar a mudança de "dicionários" de códigos —, os ganhos da compactação seriam mínimos, pois, com a mudança constante de dicionários, o número de zeros acabaria se tornando muito elevado. Portanto, para a codificação com um número menor de bits, aquele esquema não serviria.



- UM SUPERCOMPRESSOR
- ESQUEMA DUPLO DE CODIFICAÇÃO
- CÁLCULO DA FREQUÊNCIA DE PARES
- DESCOMPRESSÃO
- O MÉTODO CHINÊS
- CODIFICAÇÃO POR DICIONÁRIOS
- DECODIFICAÇÃO
- VANTAGENS E DESVANTAGENS

ESQUEMA DUPLO DE CODIFICAÇÃO

É possível empregar, porém, um método mais elaborado, que adota um esquema duplo de codificação:

- quanto mais freqüente a letra no texto a ser codificado, menor é o número de bits usado para seu código. Por exemplo: o espaço (caractere mais freqüente em qualquer texto) tem um código binário de três bits (011); a letra E, de cinco bits (10111). Já a letra P, não tão frequente, tem um código de seis bits (110110), e a letra Z, de oito bits (11111100);
- certos *pares* de letras muito comuns em um texto também recebem um código binário reduzido. O esquema de codificação é o mesmo das letras, mas o efeito de compressão é bem mais poderoso, já que estamos substituindo dois bytes (dezesseis bits) por um número muito menor de bits no novo código.

CÁLCULO DA FREQÜÊNCIA DE PARES

Apresentamos, a seguir, um programa destinado a computar e exibir os pares de letras mais frequentes em um texto. Ele usa a mesma técnica do programa que determina a frequência simples dos caracteres, fornecido no artigo anterior.

Em primeiro lugar, o programa inicializa os conjuntos **L** — que conterá a freqüência de pares que começa com cada caractere ASCII, de 32 a 90 — e **F** — que conterá em cada casela **F(I,J)** a freqüência acumulada pelo par de caracteres com códigos **I** e **J**.

Em todas as versões, exceto a do Spectrum, o sinal % depois de **L** e **F** serve para definir como inteiro o tipo do conjunto. Isso economiza memória e aumenta a velocidade de processamento. As linhas 20 a 40 não são necessárias nos computadores que inicializam em 0 todas as variáveis numéricas, quando **RUN** é digitado.



```
10 DIM L%(60),F%(60,60),C%(60)
20 NT=0:NP=0
30 FOR I=1 TO 60:L%(I)=0
35 FOR J=1 TO 60
40 F%(I,J)=0:NEXT J:NEXT I
```

Coloque um comando **CLEAR 3000** na linha 10, antes da declaração **DIM**, para o TRS-80 e TRS-Color.

```
10 DIM L(60),f(60,60),c(60)
20 LET nt=0·LET np=0
30 FOR i=1 TO 60:LET L(i)=0
35 FOR j=1 TO 60
40 LET f(i,j)=0:NEXT j:NEXT i
```

A parte seguinte do programa principal lê as linhas **DATA** que contêm o texto a ser codificado. Para fins de teste, você poderá recorrer ao texto de instruções de uma aventura, dado no artigo anterior, acrescentando-o ao final deste programa.



```
60 PRINT "ANALISANDO..."
70 READ L$:IF L$="*" THEN 100
75 PRINT L$:LT=LEN(L$)
80 NT=NT+LT
85 IF INT(LT/2)<>LT/2 THEN L$=
L$+" "
90 GOSUB 520·GOTO 70
```

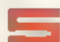
```
60 PRINT "ANALISANDO   "
70 READ L$·IF L$="*" THEN GOTO
100
75 PRINT L$·LET Lt=LEN L$
80 LET nt=nt+lt
85 IF INT Lt/2 <>Lt/2 THEN LET
L$=L$+" "
90 GOSUB 520.GOTO 70
```

Um asterisco na linha de texto (**L$**) indica o fim do mesmo. Caso o texto não tenha terminado, calcula-se o comprimento **LT** de **L$**. Se **LT** for ímpar, é preciso acrescentar um espaço em branco ao final de **L$**, para que a rotina de contagem efetue corretamente o cálculo e a extração de pares de caracteres. Isso é feito pela linha 85, que verifica se o resto da divisão de **LT** por 2

é maior que 0 (se for, é ímpar). Finalmente, a linha 90 chama a rotina de contagem, que começa em 520:



```
500 REM - ROTINA DE CONTAGEM
520 FOR I=1 TO LEN(L$)-1
525 NP=NP+1
530 C1=ASC(MID$(L$,I,1))-31
540 C2=ASC(MID$(L$,I+1,1))-31
545 L%(C1)=L%(C1)+1
550 F%(C1,C2)=F%(C1,C2)+1
560 NEXT I
570 RETURN
```



```
500 REM - ROTINA DE CONTAGEM
520 FOR i=1 TO LEN L$-1
525 LET np=np+1
530 LET c1=ASC(L$(i TO i))-31
540 LET c2=ASC(L$(i+1 TO i+1))-31
545 LET L(c1)=L(c1)+1
550 LET f(c1,c2)=f(c1,c2)+1
560 NEXT I
570 RETURN
```

A rotina de contagem é muito simples: o laço que vai da linha 520 à 560 percorre **L$** tomando um caractere por vez. As variáveis **C1** e **C2** conterão os códigos ASCII do primeiro e do segundo caracteres de um par. Subtraindo o valor 31 desse código, obteremos um número compreendido entre 1 (espaço) e 58 (letra Z). **C1** e **C2** servirão assim como índices para acumular a casela correta de **F** e de **L**.

O programa principal termina quando se atinge o final do texto e a rotina dos resultados é chamada.



```
100 PRINT "TOTAL: ";NT;"CARACTERES E ";NP;"PARES"
120 GOSUB 780
140 STOP
```

A rotina de exibição, que começa em 780, utiliza uma rotina de ordenação para mostrar os resultados em ordem decrescente, ou seja, os pares mais freqüentes em primeiro lugar.



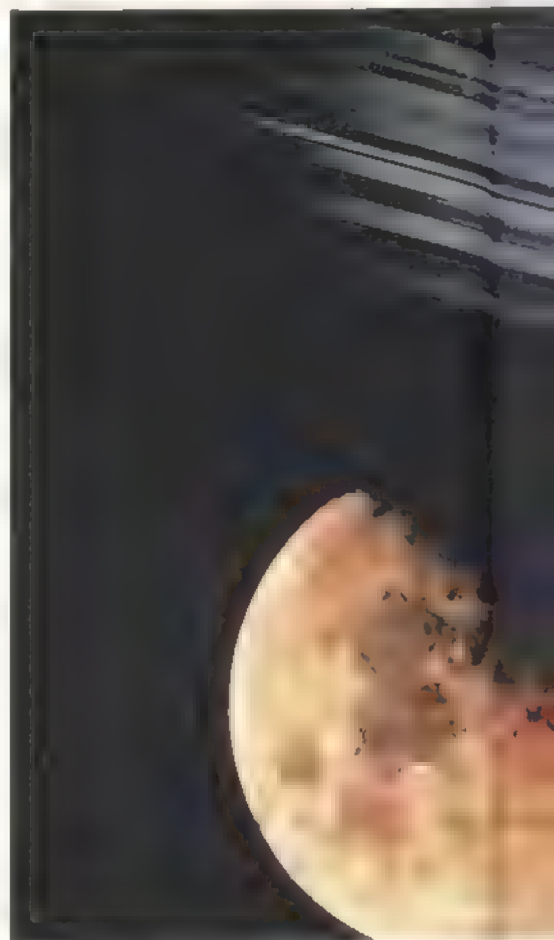
```
680 REM - ROTINA DE ORDENACAO
690 N=60
695 FOR J=1 TO 60:C%(J)=J:NEXT
700 FL=0
710 N=N-1:FOR J=1 TO N
720 IF F%(I,C%(J))>=F%(I,C%(J+1)) THEN GOTO 740
730 X=C%(J):C%(J)=C%(J+1):C%(J+1)=X:FL=1
740 NEXT J
750 IF FL=0 OR N=2 THEN RETURN
760 GOTO 700
770 REM - ROTINA DE IMPRESSAO
780 NL=0:PRINT
790 PRINT "FREQUENCIA DE PARES NO TEXTO":PRINT
800 FOR I=1 TO 60:IF L%(I)=0 THEN 850
805 GOSUB 690
810 FOR J=1 TO 60
815 X=C%(J):IF F%(I,X)=0 THEN GOTO 825
820 PRINT CHR$(I+31)+CHR$(X+31);F%(I,X);" ";
825 NEXT J:PRINT
830 NL=NL+1:IF NL<15 THEN GOTO 850
840 NL=0:INPUT "PRESSIONE <ENTER> ";X$
850 NEXT I:PRINT
860 RETURN
```



```
680 REM - ROTINA DE ORDENACAO
690 LET n=60
695 FOR j=1 TO 60:LET c(j)=j
700 NEXT j:LET fL=0
710 LET n=n-1:FOR j=1 TO n
720 IF f(i,c(j))>=f(i,c(j+1)) THEN GOTO 740
730 LET x=c(j):LET c(j)=c(j+1):LET c(j+1)=x:LET fL=1
740 NEXT j
750 IF fL=0 OR n=2 THEN RETURN
760 GOTO 700
770 REM - ROTINA DE IMPRESSAO
780 LET nL=0:PRINT
790 PRINT "FREQUENCIA DE PARES NO TEXTO":PRINT
800 FOR i=1 TO 60:IF L(i)=0 THEN GOTO 850
805 GOSUB 690
810 FOR j=1 TO 60
815 LET x=c(j):IF f(i,x)=0 THEN GOTO 825
820 PRINT CHR$ (i+31)+CHR$ (x+31);f(i,x);" ";
825 NEXT j:PRINT
830 LET nL=nL+1:IF nL<15 THEN GOTO 850
840 LET nL=0:INPUT "PRESSIONE ENTER> ";x$
850 NEXT i:PRINT
860 RETURN
```

Para não alterar a matriz **F**, a rotina de ordenação utiliza um conjunto **C**, que funciona como índice — para isso, ele é inicializado no começo da rotina (linha 695), de modo a conter os números 1, 2, 3 etc. A ordenação (do tipo bolha) coloca esses números em uma ordem diferente.

A sub-rotina de impressão examina cada uma das linhas da matriz **F**. Se o total **L(I)** da linha **I** for 0, não se registrou a ocorrência de nenhum caractere



com código **I+31** no texto. Do contrário, o programa chama a rotina de ordenação. O laço que vai da linha 810 à 825 imprime todos os pares cuja freqüência é maior que 0. A variável-índice **X**, extraída de **C**, é usada para mostrá-los em ordem decrescente.

Se você quiser ver apenas os dois pares mais freqüentes para cada caractere, modifique a linha 810 para:

```
810 FOR J=1 TO 2
```

O MÉTODO CHINÊS

A técnica de compressão explicada a seguir apresenta um alto nível de eficiência, dependendo do texto a codificar e de sua extensão. Trata-se, na realidade, de uma generalização do algoritmo de codificação por pares de letras: se podemos empregar códigos numéricos para representar os pares de caracteres mais freqüentes em um texto, a fim de comprimi-lo, é possível também usar tríades dos caracteres mais comuns. Naturalmente, essa abordagem se tornaria cada vez mais inviável, à medida que aumentássemos as dimensões da matriz

F: de 3.600 elementos, para contagem de pares, para 216.000 elementos, para contagem de tríades etc.

E por que não montar um dicionário com palavras completas? Cada palavra receberia um código numérico de oito ou de dezesseis bits, e o texto seria composto por uma sequência desses códigos. Para reconstituí-lo, bastaria buscar no dicionário a palavra correspondente a cada número.

Esse sistema é chamado método chinês, pois as palavras em chinês não são formadas a partir de um pequeno conjunto de fonemas ou sílabas, como nos idiomas ocidentais, mas de figuras únicas, os *ideogramas*, que correspondem a códigos.

A idéia pode ser muito boa para uma linguagem natural, mas não devemos nos esquecer de que, no computador, o dicionário também precisa ser armazenado. Assim, para se conseguir o efeito de compressão, é necessário que o resultado da soma do número de bytes obtido no processo com o número de bytes gasto com a armazenagem do dicionário na memória seja menor do que o número de bytes do texto não comprimido (original). A taxa ou eficiência de compressão é dada pela divisão de um valor pelo outro.

Para entender como funciona esse algoritmo e verificar se é capaz de comprimir o texto-exemplo usado antes, digite e execute este programa.



```
10 DIM F(200),C(200),P$(200),F$
(50)
20 NT=0:NP=0:NR=0:L
ET NL=0:NF=0
30 FOR I=1 TO 200
40 F(I)=0:C(I)=I
50 NEXT I
60 PRINT "ANALISANDO..."
70 READ L$:IF L$ "*" THEN GOTO
100
75 NL=NL+1
80 NT=NT+LEN(L$)
85 GOSUB 510
90 NF=NF+LEN(F$(NL)):GOTO 70
100 PRINT "TOTAL. ",NT;"CARACTE
RES E ";NP;"PALAVRAS"
110 PRINT "DICIONARIO COM ";NR,
"CARACTERES."
115 PRINT "TEXTO COMPRIMIDO COM
 ";NF;"CARACTERES"
120 PRINT "ORDENANDO.  "
130 GOSUB 690:GOSUB 780
250 STOP
500 REM - ROTINA DE CONTAGEM
510 L$=L$+" ":I=1:L=1
520 C=ASC(MID$(L$,I,1))
525 IF C<>32 THEN GOTO 560
540 I=I+1:IF I<LEN(L$) THEN
 GOTO 520
550 RETURN
560 L=I
570 C=ASC(MID$(L$,I,1))
575 IF C=32 THEN GOTO 610
590 I=I+1:IF I=<LEN(L$) THEN
 GOTO 570
600 RETURN
610 X$=MID$(L$,L,I-L)
620 FOR N=1 TO NP
630 IF X$<>P$(N) THEN GOTO 640
635 F(N)=F(N)+1:F$(NL)=F$(NL)+
 CHR$(N):GOTO 520
640 NEXT N
650 NP=NP+1:F(NP)=1:P$(NP)=X$
660 NR=NR+LEN(X$):F$(NL)=F$(NL)
 +CHR$(NP)
670 GOTO 520
680 REM - ROTINA DE ORDENACAO
690 N=NP
700 FL=0
710 N=N-1:FOR I=1 TO N
720 IF F(C(I))=>F(C(I+1)) THEN
GOTO 740
730 X=C(I):C(I)=C(I+1)
735 C(I+1)=X:FL=1
740 NEXT I
```

```
750 IF FL=0 OR N=2 THEN RETURN
760 GOTO 700
770 REM - ROTINA DE IMPRESSAO
780 NX=0
790 PRINT "FREQUENCIA DE PALAVR
AS NO TEXTO":PRINT
800 FOR I=1 TO NP
820 PRINT I;P$(C(I)),F(C(I))
830 NX=NX+1:IF NX<15 THEN GOTO
 850
840 NX=0:INPUT "PRESSIONE <ENTE
R> ";X$
850 NEXT I:PRINT
860 RETURN
```

Os usuários do TRS-80 e do TRS-Color devem colocar um comando **CLEAR 3000** antes do **DIM**, na linha 10.

```
10 DIM f(200),c(200),p$(200,15)
,f$(50)
20 LET nt=0:LET np=0:LET nt=0:L
ET nl=0:LET nf=0
30 FOR i=1 TO 200
40 LET f(i)=0:LET c(i)=i
50 NEXT i
60 PRINT "ANALISANDO..."
70 READ L$:IF L$="*" THEN GOTO
100
75 LET nl=nl+1
80 LET nt=nt+LEN L$
85 GOSUB 510
90 LET nf=nf+LEN f$(nl):GOTO 70
100 PRINT "TOTAL: ";nt;"CARACTE
RES E ";np;"PALAVRAS"
110 PRINT "DICIONARIO COM ";nr;
"CARACTERES."
115 PRINT "TEXTO COMPRIMIDO COM
 ";nf;"CARACTERES"
120 PRINT "ORDENANDO..."
130 GOSUB 690:GOSUB 780
250 STOP

500 REM - ROTINA DE CONTAGEM
510 LET L$=L$+" ":LET i=1:LET L
=1
520 LET c=ASC L$(i TO i)
525 IF c<>32 THEN GOTO 560
540 LET i=i+1:IF i<LEN L$ THEN
 GOTO 520
550 RETURN
560 LET L=i
570 LET c=ASC L$(i TO i)
575 IF c=32 THEN GOTO 610
590 LET i=i+1:IF i=<LEN L$ THEN
 GOTO 570
600 RETURN
610 LET x$=L$(L to I-1)
620 FOR n=1 TO NP
630 IF x$<>p$(n) THEN GOTO 640
635 LET f(n)=f(n)+1:LET f$(nl)=
f$(nl)+CHR$(n):GOTO 520
640 NEXT n
650 LET np=np+1:LET f(np)=1:LET
p$(np)=x$
660 LET nr=nr+LEN x$:LET f$(nl
)=f$(nl)+CHR$(np)
670 GOTO 520
680 REM - ROTINA DE ORDENACAO
690 LET n=np
700 LET fl=0
710 LET n=n-1:FOR i=1 TO n
720 IF f(c(i))=>f(c(i+1)) THEN
GOTO 740
730 LET x=c(i):LET c(i)=c(i+1):
LET c(i+1)=x:LET fl=1
740 NEXT i
750 IF fl=0 OR n=2 THEN RETURN
760 GOTO 700
770 REM - ROTINA DE IMPRESSAO
780 LET nx=0
790 PRINT "FREQUENCIA DE PALAVR
AS NO TEXTO":PRINT
800 FOR i=1 TO np
820 PRINT i;p$(c(i)),f(c(i))
830 LET nx=nx+1:IF nx<15 THEN G
OTO 850
840 LET nx=0:PRINT "PRESSIONE <
ENTER> ":INPUT x$
850 NEXT i:PRINT
860 RETURN
```

A parte principal do programa é a rotina de extração de palavras, que vai da linha 500 à 670.

Para identificar e separar as palavras contidas em uma linha de texto, essa rotina procura o primeiro caractere não-branco (que não é um espaço) a partir do caractere I da linha **L$**. A posição inicial da palavra é armazenada na variável **L** (linha 560).

Em seguida, a rotina procura o fim da palavra: continua a percorrer o texto até encontrar o primeiro caractere de espaço. Isso é feito pelas linhas 570 a 600. Note que, para evitar que a última palavra de um texto se perca, a linha 510 da rotina sempre acrescenta um espaço em branco após **L$**.

Achado o final da palavra, o *substring* compreendido entre **L** e **I-1** (a nova posição) é copiado em **X$** (linha 610). As linhas 620 a 660 verificam então se a palavra **X$** existe no dicionário **P$**, com **NP** palavras. Se não existe, a nova palavra é acrescentada. Antes de voltar para a linha 520, para buscar nova palavra no texto, a rotina incrementa em **F** a freqüência de ocorrência da palavra. A linha de saída, que contém o texto comprimido (armazenada em **F$**), recebe um novo caractere, que corresponde ao código da palavra no dicionário. Portanto, uma palavra inteira é substituída por um byte.

Concluindo o programa, as rotinas das linhas 690 e 780 colocam as palavras do dicionário em ordem alfabética, exibindo-as na tela em grupos de quinze, como mostramos a seguir:

```
FREQUENCIA DAS PALAVRAS

DE        15
O         10
A          8
E          8
UM         6
DO         5
SUA        5
...       ..
```

Observamos no texto-exemplo que:
- as palavras curtas aparecem mais;
- há um grande número de palavras com uma única ocorrência;
- vírgulas e outros sinais são tratados como parte da palavra junto à qual se encontram.

Com isso, a eficiência de compressão do algoritmo é muito baixa: de 1228 caracteres do texto original, conseguimos comprimir apenas 215 caracteres (uma taxa de 82,5%); porém, o dicionário ocupa outros 870 bytes, dando um total de 1085. A eficiência se reduz, assim, a apenas 11,6%.

O algoritmo é mais eficaz quando se combinam textos bem longos e uso de um vocabulário reduzido.

DECODIFICAÇÃO

Para ver como o texto comprimido é reconstituído, acrescente a sub-rotina de decodificação:



```
140 GOSUB 900
890 REM - ROTINA DECODIFICACAO
900 NX=0:FOR I=1 TO NL
910 FOR J=1 TO LEN(F$(I))
920 PRINT " ",P$(ASC(MID$(F$(I)
,J,1)));
930 NEXT J:PRINT
940 NX=NX+1:IF NX<15 THEN GOTO
 950
945 NX=0:INPUT "PRESSIONE <ENTE
R> ";R$
950 NEXT I:RETURN
```

```
140 GOSUB 900
890 REM - ROTINA DECODIFICACAO
900 LET nx=0:FOR i=1 TO nL
910 FOR j=1 TO LEN f$(i)
920 PRINT " ";p$(ASC(F$(i,j TO
j)));
930 NEXT j:PRINT
940 LET nx=nx+1:IF nx<15 THEN G
OTO 950
945 LET nx=0:PRINT "PRESSIONE <
```

Essa rotina funciona de modo oposto ao da de codificação, mas é bem mais simples. Os códigos armazenados em **F$** são recuperados pela função **ASC** da linha 920, e servem como índice do dicionário **P$** para reconstruir o texto original. Os espaços entre palavras são inseridos automaticamente.

ROTINAS EM CÓDIGO DE MÁQUINA (2)

- ROTINAS NAS LINHAS DATA
- A ESTRUTURA DE UMA LINHA
- PEEK E POKE
- ENTENDA COMO FUNCIONA

A linha DATA é um excelente lugar para se colocar programas em linguagem de máquina embutidos em um programa em BASIC. Aprenda aqui os truques que facilitam essa tarefa no MSX.



Com frequência, armazenamos, em algum lugar da memória, uma rotina em código de máquina, um arquivo de padrões de um desenho ou até mesmo uma composição musical acompanhada de instruções. Muitas vezes é útil incorporar esse arquivo de rotinas a um programa em linguagem BASIC. Se colocarmos seus códigos nas linhas **DATA** do programa que os acessa, tornaremos bem mais fácil o manuseio do sistema, viabilizando a obtenção de uma listagem completa e evitando repetidas buscas no gravador ou no drive, para uma posterior junção. Com essas linhas **DATA**, precisaremos apenas acrescentar um laço que, por intermédio do comando **POKE**, carregue uma determinada região da memória com os dados nela contidos.

INTRODUÇÃO DAS LINHAS DATA

Para apresentar os códigos do arquivo, o mais conveniente é utilizar números hexadecimais, que ocupam menos espaço e reduzem a possibilidade de se cometer erros na digitação. Além disso, o modo como dispomos os números na linha pode facilitar correções ou consultas posteriores — ou seja, a quantidade de números por linha indica que ali se encontra um determinado padrão gráfico ou que aqueles códigos representam uma pequena sub-rotina dentro da rotina principal.

O programa a seguir irá auxiliá-lo nessa tarefa. Inicialmente, você irá complementá-lo com linhas **DATA** seguidas de caracteres que depois serão superpostos pelo arquivo. É importante ressaltar que a quantidade de caracteres por linha determinará o número de código nela armazenados. Assim, você pode "moldar" as linhas **DATA**, dando-lhes a aparência que julgar melhor. Por exemplo, uma linha do tipo:

```
330 DATA XXXXXXXX
```

seria transformada em:

```
330 DATA A3,F2,01
```

Convém, portanto, que você gaste algum tempo na criação dessas linhas, se quiser uma listagem bem organizada.

RODANDO O PROGRAMA

Após o acréscimo das linhas **DATA**, passamos à execução do programa. Em primeiro lugar, ele solicita o endereço inicial do arquivo, pedindo, em seguida, seu comprimento. Depois de termos digitado essas informações, o computador pergunta se as linhas **DATA** já foram introduzidas. Caso isso ainda não tenha sido feito, o programa será interrompido. Finalmente, ele pede o número da primeira linha a ser preenchida. Se não encontrar essa linha, o computador interromperá a execução do programa e imprimirá uma mensagem de erro. O mesmo irá ocorrer se a linha indicada não contiver o comando **DATA**.

É importante que todas as linhas reservadas para os códigos estejam em seqüência na listagem — durante a execução, uma mensagem de erro pára o programa quando o computador encontra uma linha que não seja **DATA**.

```
10 CLS
20 INPUT"Qual o endereço do arq
uivo ";EN:PRINT
30 INPUT"Qual o comprimento ";C
O:PRINT
40 PRINT"Você já introduziu as
linhas DATA? S/N ":PRINT
50 G$=INKEY$:IF G$="" THEN GOTO
 50
60 IF G$="N" OR G$="n" THEN PRI
NT "Eu vou parar o programa par
a você fazerisso":STOP
70 INPUT"Qual a primeira linha
DATA ";LD:PRINT
80 X=32769!
90 IF PEEK(X+2)+256*PEEK(X+3)=L
D THEN 130
100 IF PEEK(X)+256*PEEK(X+1)=0
THEN PRINT"Não achei esta linha
":STOP
110 X=PEEK(X)+256*PEEK(X+1)
120 GOTO 90
130 IF PEEK(X+4)<>132 THEN PRIN
T"Esta não é uma linha DATA":PR
INT:GOTO 70
140 X=X+3
150 FOR I=0 TO CO-1
160 X=X+3
170 IF PEEK(X)=0 THEN X=X+5:GOS
UB 310
180 IF PEEK(X+1)=0 THEN POKE X,
32:X=X+6:GOSUB 310
190 LET A$=HEX$(PEEK(E+I))
200 IF LEN(A$)=1 THEN A$="0"+A$
210 POKE X,ASC(MID$(A$,1,1))
220 POKE X+1,ASC(MID$(A$,2,1))
230 IF PEEK(X+2)=0 THEN X=X-1:G
OTO 270
240 IF PEEK(X+3)=0 THEN POKE (X
+2),32:GOTO 270
250 IF PEEK(X+4)=0 THEN POKE (X
+2),32:POKE (X+3),32:X=X+1:GOTO
 270
260 IF I<>CO-1 THEN POKE(X+2),4
4
270 NEXT I
280 IF PEEK(X+2)<>0 THEN POKE(X
+2),32:X=X+1:GOTO 280
290 LIST
300 STOP
310 IF PEEK(X)<>132 THEN PRINT"
Faltam linhas DATA":STOP ELSE X
=X+2
320 RETURN
```

LEITURA DA MEMÓRIA

Para entender o funcionamento do programa, é necessário conhecer a estrutura de uma linha **DATA**, ou seja, saber como ela é armazenada na **RAM**.

Ao ligar o computador, digite:

```
10 DATA A,B,C
```

Agora, para "ler" o que está escrito na memória, usaremos o comando **PEEK**, que mostra o conteúdo de um determinado endereço. Começaremos pelo exame do byte de número 32769, que contém a primeira informação da linha inicial de um programa em BASIC. Para isso, digite o comando direto:

```
FOR I=32769 TO 32782:PRINT PEEK
(I);" ";:NEXT I
```

Você deve ter obtido a seguinte seqüência de números:

```
13 128 10 0 132 32 65 44 66 44
67 0 0 0
```

ESTRUTURA DA LINHA

Os dois primeiros números referem-se ao endereço inicial da próxima linha. Esse endereço é armazenado na forma **LH** (do inglês Low, parte baixa, e **High**, parte alta). Pode ser decodificado do seguinte modo: **128*256+13** = 32781. O par seguinte equivale ao número da linha, que é armazenado da mesma forma: **0*256+10** = 10. O próximo número, 132, é o código da palavra **DATA**.

O MSX atribui um número — ou *token* (símbolo, indicação) — a cada palavra e caractere reservados. Assim, o computador economiza bastante memória: em vez de armazenar todos os caracteres da palavra, guarda apenas o seu token. É importante não confundir os tokens com os códigos ASCII, que representam os caracteres comuns.

O número 32, que vem a seguir, é o código ASCII do espaço em branco, que separa os dados do comando **DATA**. Os próximos cinco bytes contêm esses dados, que vêm separados por uma vírgula (44). O primeiro 0, na seqüência, indica o fim de uma linha, e os dois outros, o fim do programa. Quando todas as linhas tiverem sido examinadas, os dois bytes iniciais apontam para o primeiro destes dois zeros.

FUNCIONAMENTO DO PROGRAMA

Após conseguir as informações solicitadas no início do programa, o computador passa às verificações.

A linha 80 inicializa a variável principal, **X**, que contém o endereço do byte em estudo. Primeiro, ela assume o valor 32769, que, conforme vimos, é o endereço do byte inicial da primeira linha. A linha 90 verifica o terceiro e quarto bytes, obtendo o número da linha. Se esta for a linha procurada (a primeira linha **DATA**), o programa é desviado para a linha 130.

A linha 100 pesquisa o primeiro e o segundo bytes, que contêm o endereço da próxima linha. Se houver um número 0 no lugar do endereço, estamos no fim do programa — o que significa que a linha não foi encontrada. O computador imprime então uma mensagem de erro. Caso contrário, é necessário pesquisar a próxima linha. Assim, depois de atribuir à variável **X** o valor encontrado no primeiro e no segundo bytes (linha 110), o programa retorna para a linha 90. Esse laço só será interrompido quando a próxima linha do programa foi localizada ou quando chegarmos ao final da listagem.

Ao encontrar a linha indicada pelo usuário, o microcomputador verifica se se trata de uma linha **DATA**. Para isso, pesquisa seu quinto byte. Se ele não contiver o token correspondente à instrução **DATA**, será impressa uma mensagem de erro.

O LAÇO PRINCIPAL

Feitas todas essas verificações, o programa inicia o laço principal, controlado pela variável **I**, que coloca os códigos de máquina no programa em BASIC. Antes, porém, a linha 140 acrescenta três unidades a **X**, para que, dentro do laço, essa variável seja incrementada em mais três e alcance o endereço do primeiro caractere da linha que deverá ser preenchida.

Primeiro, o laço principal busca o código na região da memória determinada pelo usuário, transforma-o na notação hexadecimal (linha 190) e o coloca sobre os caracteres para os quais reservamos espaço nas linhas **DATA** (linhas 210 e 220). Se esse código for inferior a 15, sua notação em hexadecimal terá apenas um caractere. Para dar uma melhor apresentação ao programa, a linha 200 acrescentará um 0 à direita desse caractere.

As linhas 170 e 180 são responsáveis pela mudança de linha. Se o byte em questão contiver o valor 0, um salto será necessário, pois esse valor indica o fim da linha. Incrementa-se então a variável **X**, de modo que ela alcance o endereço do próximo caractere a ser superposto por um código, na linha seguinte. Porém, se o byte posterior a este contiver um 0, não será possível colocar um código na linha, por falta de espaço. Nesse caso, além do incremento, é preciso imprimir um espaço em branco sobre o caractere que se encontra no endereço anterior ao 0. Ambas as linhas, antes do salto, utilizam uma sub-rotina (linha 310) para verificar se a próxima linha é realmente uma linha **DATA**.

Convém estar atento para que nada seja impresso após o último código, pois ele deve finalizar a linha **DATA**. Assim, quando o código já foi introduzido nas posições **X** e **X+1**, não haverá vírgula se o primeiro, o segundo ou o terceiro bytes após ele contiverem o número 0. Nesse caso, alguns espaços em branco serão colocados sobre os caracteres anteriores àqueles bytes. O valor de **X** também sofrerá uma pequena mudança para que, na próxima volta do laço, o endereço **X**, incrementado em três, corresponda ao endereço de 0, e a linha 170 se encarregue de promover o salto para a próxima linha **DATA**.

Se o primeiro, o segundo e o terceiro bytes após o código não contiverem 0, a linha 260 se encarregará de imprimir uma vírgula.

Quando o programa sai do laço, a linha 280 limpa a última linha **DATA**, imprimindo espaços em branco sobre os caracteres que ali se encontravam.

Finalmente, o computador exibe na tela a listagem completa, mostrando ao usuário os códigos já incorporados ao programa.

Como estamos fazendo modificações dentro do programa, convém gravá-lo antes da execução. Um pequeno erro de digitação pode provocar um **POKE** e inutilizar o programa.

LISTA DE TOKENS

Com os novos conhecimentos sobre a armazenagem das linhas em BASIC, você pode fazer uma série de experiências com o comando **POKE**. Tente, por exemplo, digitar uma linha e mudar o seu número com um **POKE** no primeiro e no segundo bytes. Lembre-se de que o primeiro byte do BASIC é o de número 32769. Você pode também "olhar" as linhas armazenadas no BASIC recorrendo a alguns comandos diretos, como fizemos em nosso exemplo.

Provavelmente, será útil dispor de uma lista dos comandos com seus respectivos tokens. Para obtê-la, digite e execute este programa:

```
10 E=14962
20 C 65
30 PRINT CHR$(C);
40 A=PEEK(E)
50 B=PEEK(E+1)
60 A$=CHR$(A)
70 IF A<128 THEN PRINT A$;:GOTO
 110
80 PRINT CHR$(A-128);TAB(8);B
90 E=E+1
100 IF PEEK(E+1)<>0 THEN PRINTC
HR$(C);
110 IF PEEK(E+1)<>0 THEN 150
120 C=C+1:IF C=89 THEN 170
125 B$=CHR$(C) PRINT
130 IF B$="J" OR B$="Q" THEN 15
0
140 PRINT B$,
150 E=E+1
160 IF E<=15649 THEN 40
170 E=15654:PRINT
180 A=PEEK(E).B=PEEK(E+1)
190 IF A>127 THEN PRINT CHR$(A-
128);
200 IF A<128 THEN PRINT CHR$(A)
;
210 PRINT TAB(8);B
220 E=E+2·PRINT
230 IF E<156/3 THEN GOTO 180
```

UM EDITOR MUSICAL (3)

Complete a listagem do seu editor musical e entregue-se ao prazer de compor e executar melodias. As instruções dadas a seguir mostram como utilizar cada função do programa.

Apresentamos aqui a última parte do editor musical. Uma vez adicionada ao restante do programa, você poderá se dedicar à composição e execução de peças musicais.

O programa foi feito de modo a obter o máximo de cada máquina, em termos de efeitos sonoros. Como os recursos das diferentes linhas de micros variam muito entre si, as versões são bem distintas, merecendo, cada qual, uma explicação. Em alguns casos, as instruções parecerão bem complicadas. Mas, seja como for, não há melhor maneira de aprender a usar o editor do que sentar-se diante do micro, carregar o programa completo e tentar entender um comando de cada vez.

S

O menu do programa do computador Spectrum oferece sete opções. Selecione a opção 1, que transforma o teclado do micro num piano.

A disposição das notas é igual à que vimos no artigo da página 721. O dó mais grave corresponde à tecla Q; o dó médio, a tecla 1, e o do mais agudo, a tecla B. O usuário deve determinar se irá acrescentar notas à melodia anterior, ou iniciar uma nova. Como nada foi composto ainda, escolha 'S' para começar. Defina também a duração de cada nota, usando <SYMBOL> <SHIFT> seguido de um número de 1 a 5 (semicolcheia, colcheia, semínima, mínima, semibreve, respectivamente — quanto maior o número, mais demorada é a nota). Pode-se estabelecer a duração de cada nota antes de digitá-la e, também, corrigi-la após digitar a melodia completa, selecionando a opção 4.

O programa emite os sons à medida que as teclas são pressionadas, armazenando as notas musicais correspondentes na memória. Depois, pode-se executar a melodia completa sempre que se desejar através da opção 3. Para modificar o andamento da melodia, usa-se um número entre 1 e 15 — quanto maior



o número, mais rápido o andamento. A opção 2 possibilita a digitação das notas por meio de um código simples. As doze notas possíveis — C (dó), C # (dó sustenido), D (ré), até o B (si) — são numeradas de 0 a 11, e a pausa recebe o código −4. A oitava da melodia é especificada por um valor compreendido entre 1 e 7, e o número que indica a duração de cada nota corresponde à sua duração teórica. Assim, uma semicolcheia vale 1; uma colcheia, 2; uma semínima, 4, uma mínima, 8 e uma semibreve, 16. Essas durações valem, igualmente, para a pausa.

O código é muito fácil de usar. Por exemplo, a nota dó, segunda oitava, semicolcheia, tem o código 1024; a nota si, quarta oitava, semibreve, é 11416. Para pausas, 404 e o intervalo de uma semínima. Não se esqueça de que cada código tem dois dígitos; assim, uma colcheia vale 02, e não 2.

Após digitar algumas notas, pressione <RETURN> para voltar ao menu; selecione a opção 3 para tocar a melodia introduzida até o momento.

Se você quiser alterar, apagar ou acrescentar algumas notas, selecione a opção 4 do menu. Inicialmente, pressione **D**, para selecionar as notas que serão modificadas. Depois, entre o número das notas e tecle **E** para editá-las. Para mudar uma nota, digite seu número, <**RETURN**> e seu código. Para inserir uma nota, aperte **X** e entre seu número. Caso queira apagar tudo e reiniciar o trabalho, selecione a opção 5.

Quando estiver satisfeito com a melodia, grave-a através da opção 6.

A opção 7 permite recuperar uma música da fita cassete.

T

Ao executar o programa, um menu com nove opções será exibido na tela. Comece com a opção 3, para tocar música diretamente no teclado.

As notas estão dispostas em duas fileiras, como vimos no artigo da página 721. O dó menor corresponde à tecla Q; o dó médio, à I, e o dó maior, à V. A barra de espaço indica uma pausa com a duração desejada. O computador registrará cada nota que você tocar, mas não em tempo real — assim, não importa que você demore mais ou menos para teclar uma nota. Também não haverá problema se algum erro for cometido: é fácil corrigir as notas depois.

Se você quiser mudar uma oitava antes de executar uma determinada nota, recorra às teclas com setas para cima e para baixo. Para alterar a duração da nota, pressione as teclas com setas para a esquerda e a direita. A oitava e a duração correntes estarão impressas no topo da tela. Enquanto você toca, as notas vão sendo exibidas com a duração e oitava respectivas.

Depois de experimentar essa maneira de entrar com as notas, selecione a opção 4 (as notas que você tocou permanecem na memória). Ela permite a entrada das notas por um processo que em certos casos pode ser bem útil. Primeiro, digite o nome da nota, de "**a**" até "**f**", para as notas naturais; **A, C, D, F, G**, para acidentes (bemóis são colocados como sustenidos; logo, Bb = A # etc.), e "**p**", para uma pausa. Em seguida, entre a oitava — um número de 1 a 6 — e a duração da nota — letras **w, h, q, e, s**, correspondentes a semibreve, mínima, semínima, colcheia e semicolcheia. Para uma nota pontuada, simplesmente adicione um ponto no final. Por exemplo: A2e. significa nota A (lá sustenido), oitava 2, colcheia pontuada; c3w significa nota C (dó), oitava 3, semibreve.

Entre as notas na ordem que quiser e ouça a música quando desejar, selecionando a opção 7 do menu principal.

Outras opções do menu são a mudança no tempo de execução e a mudança geral de oitava (com efeito retroativo) — para executá-las, basta que teclemos um novo valor. O tempo de execução varia de 0 a 255 e indica a velocidade com que as notas são tocadas. Utilizando as opções Superior e Inferior, você pode subir ou descer a oitava de qualquer grupo de notas.

Para editar a melodia, selecione a opção 6. Com ela, você lista as notas e identifica as que quer alterar. É possível escolher entre **apagar, inserir, alterar** ou **continuar**. Experimente cada uma dessas opções. Para apagar notas, entre o número inicial e o número de notas que deseja eliminar. Para inserir, digite o número da nota que está antes do ponto onde se iniciará a inserção e entre uma nota por vez. Para alterar uma nota, tecle seu número e entre o novo conteúdo.

Finalmente, quando estiver satisfeito com a melodia, armazene-a em fita com a opção 2 e recupere-a quando quiser com a opção 1.

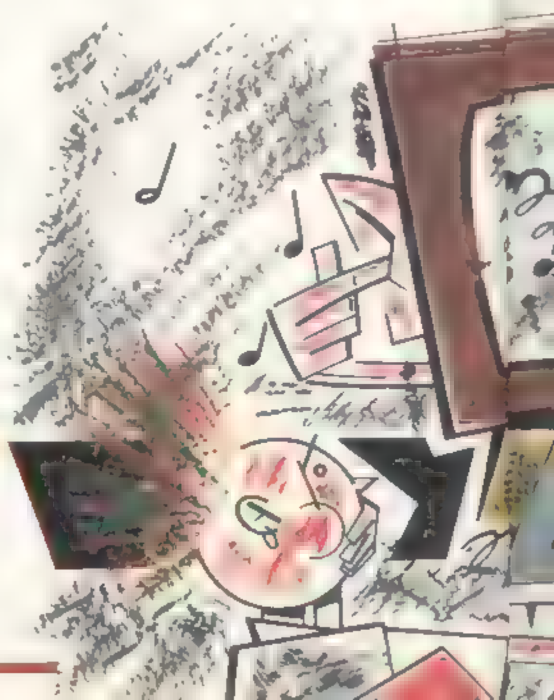
Ao iniciar a execução do programa, um menu com nove opções será exibido na tela. Comece com a opção 3 para tocar música diretamente no teclado. Teremos três oitavas disponíveis.

As notas estão dispostas no teclado em duas fileiras. **Q** corresponde ao dó da oitava mais baixa; **I**, ao dó da oitava média, e **V**, ao do da oitava mais alta. A nota A # (A ou lá sustenido) não pode ser obtida com a tecla "^", já que esta não gera nenhum código ASCII que identifique o seu pressionamento. Assim, A # foi atribuída à tecla "[", deixando uma pequena descontinuidade no teclado. A barra de espaço indica uma pausa com a duração especificada.

O computador registrará a nota que você tocar, mas não imediatamente, devido ao tempo necessário para o processamento das instruções. Portanto, não tem nenhuma importância que você demore mais ou menos para teclar uma nota. Também não haverá problema se algum erro for cometido: será fácil corrigir as notas mais tarde.

Caso você queira mudar a oitava antes de executar uma nota, use as teclas com setas para cima e para baixo. Para alterar a duração da nota, pressione as teclas com setas para a esquerda e a direita. A oitava mais baixa disponível no teclado estará impressa no topo da tela, bem como a duração das notas, que vão sendo exibidas à medida que você as toca. Para obter uma duração pontuada, selecione as teclas <**INS**> e <**DEL**>; para apagar a última nota executada, a tecla <**CLS**>.

Tendo experimentado essa forma de composição, use a tecla <**ENTER**> e volte ao menu principal. Selecione a opção 4 (as notas que você tocou permanecem na memória). Ela permite a entrada das notas por um processo que, em certos casos, pode ser muito útil. Primeiro, digite o nome da nota, de



"a" até "f", para as notas naturais; **A, C, D, F, G,** para acidentes (bemóis são colocados como sustenidos; logo, Bb = A # etc.), e "**p**", para uma pausa. Em seguida, entre a oitava — um número de 1 a 8 — e a duração da nota — letras **w, h, q, e, s, t, u**, correspondentes à semibreve, mínima, semínima, colcheia, semicolcheia, fusa e semifusa. Para uma nota pontuada, adicione um ponto no final. Por exemplo: c8t. significa C (dó), oitava 8, fusa pontuada; A4u significa A (lá sustenido), oitava 4, semifusa.

Entre as notas na ordem que quiser e ouça a música quando desejar, selecionando a opção 7 do menu principal.

Outras opções que temos são a mudança no tempo de execução das notas e a mudança geral da oitava. Esta última tem efeito retroativo, alterando a oitava de notas na memória. Para ambas as opções, basta teclar o novo valor. O tempo de execução varia de 32 a 255 e indica a velocidade com que as notas são tocadas. Com as opções Superior e Inferior você pode subir ou descer a oitava de um grupo de notas.

Para editar a melodia, selecione a opção 6. Com ela, você lista as notas e verifica o que deseja alterar. É possível escolher entre **apagar, inserir, alterar ou continuar**. Experimente cada uma dessas opções. Para apagar algumas notas, entre o numero da primeira nota e o número de notas que deseja eliminar. Para inserir, digite o número da nota imediatamente anterior ao ponto onde pretende iniciar a inserção e entre uma nota de cada. Para alterar uma nota, tecle seu número e entre o novo conteúdo.

Finalmente, quando estiver satisfeito com a melodia, armazene-a em fita com a opção 2 e recupere-a quando quiser com a opção 1.



S

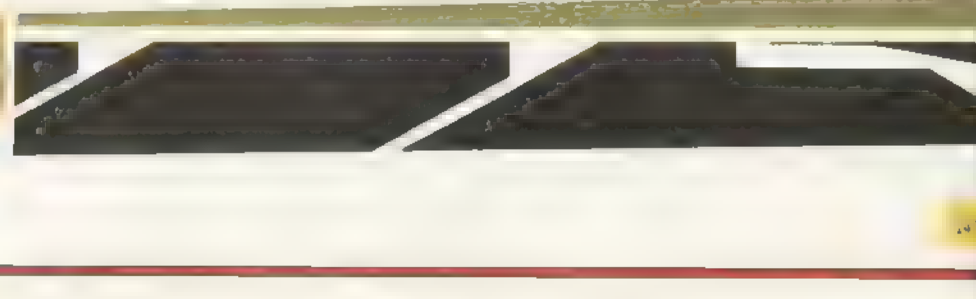
```
4452 INPUT "Entre com numero da
 nota      ";NN
4454 IF (NN<1) OR NN>ct THEN  G
OTO 4000
4460 PRINT   PRINT "Re entrando
 nota ",NN
4470 PRINT   PRINT
4480 INPUT "Entre com nova nota
 -   ";N$
4490 IF N$="" THEN  GOTO 4300
4500 FOR i=1 TO LEN (N$)  IF (N
$(i)<"0" OR N$(i)>"9") AND (N$(
i)<>"-") THEN  GOTO 4000
4510 LET N=VAL (N$)
4520 IF INT (N/1000)>11 THEN  G
OTO 4000
4530 IF N<0 THEN  GOTO 4590
4540 LET M=INT (N/100)  LET D=N
-M*100
4550 LET O=M-INT (M/10)*10: IF
O<1 OR O>7 THEN  GOTO 4000
4560 LET M=INT (M/10)+(O-1)*12
-16
4570 LET t(2*NN-1)=D: LET t(2*N
N)=M
4580 GOTO 4000
4590 LET M=INT (N/100)+1: LET D
=0-(N-M*100)
4600 IF M<>-4 THEN  GOTO 4000
4610 GOTO 4570
4700 CLS
4705 PRINT "Entre com numero da
 nota ANTES""da nova nota a se
r inserida "''"(0 para sair)"
4730 INPUT is
4735 IF is=0 THEN  GOTO 4000
4740 IF is>ct+1 THEN  GOTO 4700
4745 CLS
4750 GOSUB 2500
4755 PRINT
4760 INPUT "Entre com nova Nota
    ",N$
4765 IF LEN (N$)=0 THEN  GOTO 4
700
4770 FOR i=1 TO LEN (N$)  IF (N
$(i)<"0" OR N$(i)>"9") AND  N$(
i)<>"-") THEN  GOTO 4700
4772 LET N=VAL (N$)
4774 IF INT (N/1000)>11 THEN  G
OTO 4700
4776 IF N<0 THEN  GOTO 4792
4778 LET M=INT (N/100): LET D=N
-M*100
4782 LET O=M-INT (M/10)*10: IF
O<1 OR O>7 THEN  GOTO 4700
4784 LET M=INT (M/10)+(O-1)*12
-16
4786 FOR i=ct TO is STEP -1: LE
T t(2*(i+1))=t(2*i): LET t(2*(i
+1)-1)=t(2*i-1): NEXT i
4788 LET t(2*is-1)=D: LET t(2*i
s)=M
4790 LET ct=ct+1: GOTO 4000
4792 LET M=INT (N/100)+1: LET D
=0-(N-M*100)
4794 IF M<>-4 THEN  GOTO 4700
4796 GOTO 4786
4800 CLS
4805 PRINT "Entre com numero da
 nota a ser  apagada."''"(0 par
a sair"'
4830 INPUT de
4835 IF de=0 THEN  GOTO 4000
4840 IF de>ct THEN  GOTO 4800
4845 FOR i=de TO ct: LET t(2*i)
=t(2*(i+1)): LET t(2*i-1)=t(2*
i+1)-1): NEXT i
4850 LET ct=ct-1
4855 GOTO 4000
4900 DATA 3,0, 11, 9,0,-6,-4, .
```

MICRO DICAS

TRANSCRIÇÃO DE PARTITURAS

Nem todos conseguem executar música simplesmente sentando-se diante do micro e usando o editor musical como um processador de textos. Embora a opção de *playback* (execução da melodia entrada com o auxílio do editor) ajude bastante, nosso programa não permite representar na tela uma partitura com os detalhes de uma aplicação mais profissional.

Assim, a solução é recorrer a partituras de músicas de fácil execução para violão, flauta, em órgão eletrônico ou piano, e transcrevê-las usando o nosso editor musical.



```
,0,1
4910 DATA 12,9,8,-8,10,0,13,0,1
5,0
4920 DATA 0,16,14,2,4, 12, 7,6,
-5,-1
4930 DATA 11,-10,7,-3,5
5000 CLS
5010 INPUT "DIGITE NOME DO ARQU
IVO ";F$: LET T(maxnotes+1)=ct:
 SAVE F$ DATA T() : RETURN
6000 INPUT "DIGITE NOME DO ARQU
IVO ";F$: LOAD F$ DATA T(): LET
 ct=T(MAXNOTES+1): RETURN
```



```
1440 FOR I=ST TO NN
1450 PRINT#C,USING"###  ",I;
1460 A$=N$(I):B$=LEFT$(A$,1)
1470 IF B$>="a"AND B$<="g"THEN
C$=CHR$(ASC(B$)-32)+" "ELSE C$=
B$+"#"
1480 IF B$="p" THEN C$="- "
1490 PRINT#C,C$,
1500 IF B$<>"p"THEN PRINT#C," O
ITAVA#",MID$(A$,2,1),ELSE PRINT
#C,STRING$(11,32),
1510 PRINT#C,"  ",LE$(INSTR(R1$
,MID$(A$,3,1))),
1520 IF MID$(A$,4,1)=" "THEN PR
INT#C," " ELSE PRINT#C
1530 LP=LP+1:IF LP=12AND C=0 TH
EN LP=0:GOSUB940:CLS
1540 NEXT:GOSUB940
1550 PRINT@448,STRING$(63,32);:
PRINT@480,"PRESSIONE QUALQUER T
ECLA/MENU":EXEC 39038:RETURN
1560 CLS
1570 PRINT@8,"MODO TOCAR NOTAS"
1580 IF NN=0 THEN RETURN
1590 PRINT:INPUT"INICIO NA NOTA
 (ENTER=1)";ST
1600 IF ST<=0 THEN ST=1 ELSE IF
 ST>NN THEN ST=NN
1610 PRINT"TEMPO=",TE
1620 PLAY"V31,T"+STR$(TE)
1630 EXEC 46481:FOR I=ST TO NN
1640 PRINT@256,"TOCANDO NOTA NU
MERO",I
1650 A$=N$(I)
1660 B$=LEFT$(A$,1):IF B$="p"TH
EN 1690 ELSE IF B$>="a" AND B$<
="g"THEN P$=CHR$(ASC(B$)-32)+";
" ELSE P$=B$+"#"
1670 PLAY"O"+MID$(A$,2,1)+"L"+L
2$(INSTR(R1$,MID$(A$,3,1)))+MID
$(A$,4,1)+P$
1680 GOTO 1700
1690 PLAY"P"+L2$(INSTR(R1$,MID$
(A$,3,1)))
1700 NEXTI
1710 RETURN
1720 CLS
1730 IF NN=0 THEN RETURN
1740 PRINT@5,"MUDANCA GERAL DE
OITAVA"
1750 PRINT@64,"OITAVA SUPERIOR
OU INFERIOR(S/I)"
1760 POKE 282,245:INPUT A$
1770 IF A$="" THEN RETURN
1780 IF A$<>"S" AND A$<>"I"THEN
1750
1790 PRINT"INICIO EM(ENTER=TUDO
)";:INPUT ST
1800 IF ST<=0THEN ST=1:EN=NN:GO
TO1840
1810 INPUT "FINAL  EM(ENTER=FIM
)";EN
1820 IF EN=0 OR EN>NN THEN EN=N
N
1830 IF STT>EN THEN ST=EN
1840 FOR I=1TO NN
1850 B$=MID$(N$(I),2,1)
1860 IF A$="I"THEN C$=CHR$(ASC(
B$)-1):IF C$="0"THEN C$="5"
1870 IF A$="S"THEN C$=CHR$(ASC(
B$)+1):IF C$="6"THEN C$="1"
1880 MID$(N$(I),2,1)=C$
1890 NEXT:RETURN
1900 CLS
1910 PRINT@4,"CARREGAR MUSICA D
A FITA"
1920 PRINT:PRINT"ESTA OPCAO IRA
 APAGAR            QUALQUER MUSI
CA NA MEMORIA       -VOCE QUER C
ONTINUAR (S/N)",
1930 POKE 282,255:INPUT A$
1940 IF A$<>"S" THEN RETURN
1950 PRINT:LINE INPUT"NOMEARQ "
;A$
1960 OPEN "I",#-1,A$
1970 INPUT#-1,T$,N$
1980 NN=VAL(N$):TE=VAL(T$)
1990 FOR I=1 TO NN
2000 INPUT#-1,N$(I)
2010 NEXT:CLOSE#-1:RETURN
2020 CLS
2030 IF NN=0  THEN RETURN
2040 PRINT@5,"SALVAR MUSICA EM
FITA"
2050 PRINT:LINE INPUT"NOMEARQ."
;A$
2060 OPEN "O",#-1,A$
2070 PRINT#-1,STR$(NN),STR$(TE)
2080 FOR I=1TO NN
2090 PRINT#-1,N$(I)
2100 NEXT:CLOSE#-1:RETURN
```



```
1610 FOR I=SL TO EL
1620 A$=N$(I):B$=LEFT$(A$,1)
```

```
1950 INPUT A$:IF A$="" THEN RET
URN
1960 IF A$<>"S" AND A$<>"I"THEN
 1940
1970 LOCATE 4,9:INPUT"INICIO EM
 (ENTER=  1 )";ST
1980 IF ST<=0 THEN ST=1:EN=NN
1990 LOCATE 4,10:INPUT"FINAL  E
M (ENTER=FIM )";EN
2000 IF EN=0 OR EN>NN THEN EN=N
N
2010 IF ST>EN THEN ST=EN
2020 FOR I=ST TO EN
2030 B$=MID$(N$(I),2,1)
2040 IF A$="I" THEN C$=CHR$(ASC
1630 IF B$>="a"AND B$<="g"THEN
C$=CHR$(ASC(B$)-32)+"    "ELSE C
$=B$+"#  "
1640 IF B$="p" THEN C$="-    "
1650 C$=" "+C$
1660 IF B$<>"p"THEN C$=C$+"OITA
VA #"+MID$(A$,2,1)ELSE C$=C$+ST
RING$(9,32)
1670 C$=C$+" "+LE$(INSTR(R1$,MI
D$(A$,3,1)))
1680 IF MID$(A$,4,1)="."THEN C$
=C$+" "ELSE C$=C$+" "
1690 IF C=1 THEN LOCATE 5,LP+5:
PRINT USING"###";I: PRINT C$ELS
E LPRINT USING"###";I;:LPRINT C
$
1700 IF RT<>0 THEN RETURN
1710 LP=LP+1:IF LP=13ANDC=1 THE
N LP=0 :GOSUB 1010:CLS
1720 NEXT:GOSUB 1010
1730 LOCATE 2,18:PRINT"APERTE Q
UALQUER TECLA-MENU PRINCIPAL"
1740 Q$=INKEY$:IF Q$="" THEN 17
40
1750 RETURN
1760 CLS:COLOR 1,10:LOCATE 8,1:
PRINT"MODO -TOCAR NOTAS"
1770 ST=1:IF NN=0 THEN RETURN
1780 LOCATE 4,5:INPUT"INICIO NA
 NOTA (ENTER=1)";ST
1790 IF ST<=0 THEN ST=1 ELSE IF
 ST>NN THEN ST=NN
1800 LOCATE 4,7:PRINT"TEMPO DE
EXECUÇAO=";TE
1810 PLAY"V15T"+STR$(TE)
1820 FOR I=ST TO NN
1830 LOCATE 6,12:PRINT"TOCANDO
NOTA:",I
1840 A$=N$(I)
1850 B$=LEFT$(A$,1):IF B$="p"TH
EN 1880 ELSE IF B$>="a"ANDB$<="
g"THEN P$=CHR$(ASC(B$)-32)+" "E
LSE P$=B$+"#"
1860 PLAY"O"+MID$(A$,2,1)+"L"+L
2$(INSTR(R1$,MID$(A$,3,1)))+P$+
MID$(A$,4,1)
1870 GOTO 1890
1880 PLAY"R"+L2$(INSTR(R1$,MID$
(A$,3,1)))+MID$(A$,4,1)
1890 NEXT I
1900 RETURN
1910 CLS:COLOR 1,3:ST=1:EN=NN
1920 IF NN=0 THEN RETURN
1930 LOCATE 7,1:PRINT"MUDANÇA G
ERAL DE OITAVA"
1940 LOCATE 0,6:PRINT"OITAVA SU
PERIOR OU INFERIOR(S/I)";
(B$)-1):IF C$="0" THEN C$="1"
2050 IF A$="S" THEN C$=CHR$(ASC
(B$)+1):IF C$="9" THEN C$="8"
2060 MID$(N$(I),2,1)=C$
2070 NEXT:RETURN
2080 CLS:COLOR 15,6
2090 LOCATE 5,1:PRINT"CARREGAR
MUSICA DO GRAVADOR";
2100 LOCATE12,3:PRINT"ATENÇÃO
"
2110 LOCATE 4,4:PRINT"  ESTA OP
ÇÃO IRÁ APAGAR              Q
UALQUER MUSICA NA MEMÓRIA"
2120 LOCATE 4,7:INPUT"VOCE QUER
 CONTINUAR? S/N ";A$
2130 IF A$<>"S" THEN RETURN
2140 LOCATE 4,10:PRINT"aperte L
OAD no gravador"
2150 LOCATE 4,12:LINE INPUT"NOM
E ARQ ";A$
2160 OPEN "CAS:A$" FOR INPUT AS
#1
2170 INPUT #1,NN,TE
2180 FOR I= 1 TO NN
2190 INPUT #1,N$(I)
2200 NEXT:CLOSE#1: RETURN
2210 CLS:COLOR 15,6
2220 LOCATE 8,1:PRINT"SALVAR MU
SICA EM FITA"
2230 LOCATE 4,6:PRINT"aperte SA
VE no gravador"
2240 LOCATE 4,8:LINE INPUT"NOME
ARQ:";A$
2250 OPEN "CAS:A$" FOR OUTPUT A
S#1
2260 PRINT #1,NN,TE
2270 FOR I= 1 TO NN
2280 PRINT #1,N$(I)
2290 NEXT:CLOSE #1:RETURN
```



SPRITES EM LOGO PARA O MSX

Algumas versões mais poderosas do LOGO permitem a programação de sprites. Seguindo as instruções deste artigo, os usuários do MSX poderão gerar e controlar quarenta figuras diferentes.

Em artigos anteriores sobre a linguagem LOGO, vimos como desenhar na tela com a tartaruga, um cursor gráfico que pode ser ativado ou desativado por meio de comandos primitivos.

Como você deve ter notado, a elaboração de uma figura mais complexa demora um tempo razoavelmente longo, o que torna impossível, por exemplo, desenvolvermos um videogame com animação gráfica — tarefa simples em BASIC ou outra linguagem imperativa. Algumas versões do LOGO, porém, permitem a geração e utilização de *sprites*.

Nos artigos das páginas 188 e 808, vimos como trabalhar com sprites no MSX, única máquina nacional que possui esse recurso implementado em hardware. O Spectrum, o TRS-Color, o Apple e o TK 2000 têm comandos especiais para definir blocos gráficos (UDG) e usá-los em animações rápidas. Mas esses micros não dispõem de sprites "verdadeiros" — ou seja, não contam com controle de hardware (pelo *Video Display Processor*) que garanta a movimentação dos sprites na tela em diferentes planos, destacando-os ou ocultando-os conforme os critérios que determinam a prioridade de uns sobre os outros.

Aproveitando os recursos do MSX, o HotLOGO, versão lançada no Brasil pela Sharp para computadores dessa linha, inclui entre suas propriedades a programação de sprites verdadeiros.

Algumas versões de LOGO para o Apple e o TRS-Color também fazem uso de sprites. No exterior, a linha Commodore 64 tem poderosos instrumentos para a programação de sprites multicoloridos. Mas, em todas essas máquinas, os sprites não são controlados por hardware, como nos micros da linha MSX.

SPRITES PRÉ-PROGRAMADOS

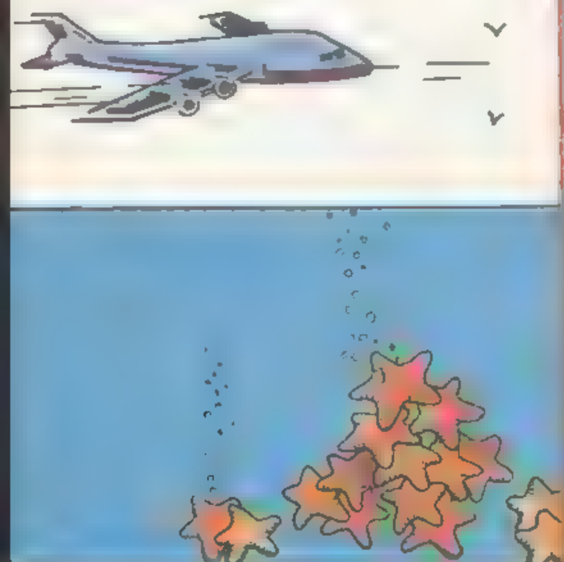
Existem dois tipos de sprites em HotLOGO: os pré-programados e os programáveis pelo usuário. Cada um deles recebe um número inteiro, entre 0 e 59.

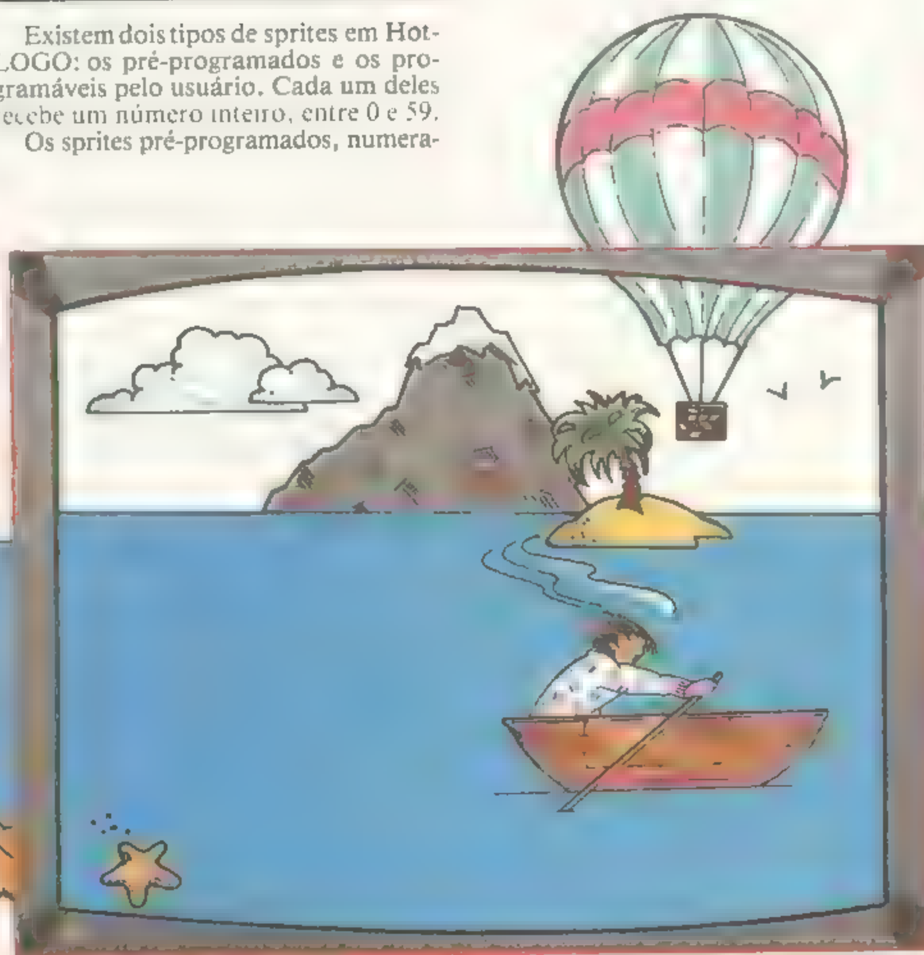
Os sprites pré-programados, numerados de 0 a 9 e de 36 a 59, são matrizes de 16 x 16 pixels contendo figuras que o usuário não pode mudar. O acesso a eles é dado pelo comando **MUDEFIG**, seguido do número do sprite.

As figuras de 0 a 9 são:

0 círculo	5 foguete
1 coração	6 tijolo
2 gato	7 helicóptero
3 cachorro	8 locomotiva
4 caminhão	9 vagão

Os sprites numerados de 36 a 59 representam tartarugas em diversos ângulos e posições. Todos aceitam os mesmos comandos e podem ser movimentados simultaneamente na tela. Os spri-





- O QUE É UM SPRITE
- TARTARUGAS X SPRITES
- VERSÕES DO LOGO COM SPRITES
- SPRITES FALSOS E VERDADEIROS
- HOTLOGO
- SPRITES PRÉ-PROGRAMADOS
- COMO PROGRAMAR UM NOVO SPRITE

tes fixos (0 a 9), porém, não mudam de orientação, como ocorre com as tartarugas ao encontrarem um comando **PARADIREITA** ou **PARAESQUERDA**.

SPRITES PROGRAMÁVEIS

Os sprites numerados de 10 a 35 podem ser programados pelo usuário.

O primitivo utilizado para desenhar um sprite novo na tela é muito simples: chama-se **EDFIG** (edita figura), e deve indicar o número do sprite a ser criado ou modificado. Quando digitamos esse comando, a tela fica em branco e aparece um cursor piscando no canto superior esquerdo. Podemos movimentá-lo em qualquer direção, por meio das teclas de controle. Para acender um pixel (bit correspondente, na matriz do sprite), pressiona-se a tecla de espaço no ponto onde estiver o cursor. Para apagá-lo, pressiona-se a mesma tecla sobre esse ponto.

Se você quiser sobrepor uma grade à tela em branco, a fim de visualizar melhor o padrão do desenho, pressione as teclas **<CTRL>** e **<K>**.

Depois de criar ou modificar a figura a seu gosto, tecle **<ESC>**: ela será armazenada na memória, sob o número indicado no comando **EDFIG**. Se não quiser guardar a figura que está editando, tecle **<CTRL>** e **<STOP>**.

ANIMAÇÃO GRÁFICA

Na produção de animações gráficas com sprites, usam-se os comandos **PARAFRENTE, PARATRÁS, PARADIREITA** e **PARAESQUERDA** para mover a figura de um lado a outro. Por exemplo:

```
RG
MUDEFIG 7
USENADA
PARAFRENTE 20
ESPERE 300
PARADIREITA 90
PARAFRENTE 90
ESPERE 300
PARATRAS 30
PARADIREITA 90 PARAFRENTE 50
```

O helicóptero (sprite pré-programado 7) parecerá levantar vôo, esperar um momento, voar para a direita, subir mais um pouco e, por fim, aterrissar.

Para simular certas ações — como um homem correndo —, podemos usar dois ou mais sprites parecidos, mostrando a figura em diferentes estágios do movimento. O comando **MUDEFIG** se encarregará de alterná-los, enquanto se provoca um deslocamento na tela.

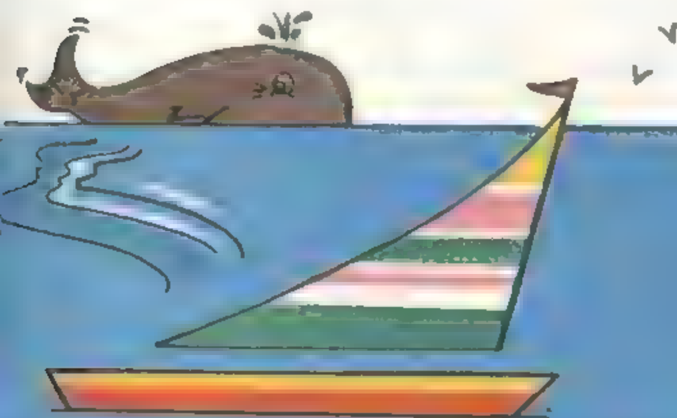


Como funciona um sprite?

A característica fundamental de um sprite é a sua velocidade. Aliás, é exatamente esta a origem do nome: *sprite*, em inglês, designa um pequeno duende que, segundo a lenda, corre como o vento.

Outra característica do sprite é a sua capacidade de interpenetração, ou seja, se duas figuras estiverem ocupando a mesma área de vídeo, aquela que for de menor prioridade dará a impressão de estar passando "por trás" da outra.

Teoricamente, pode-se reproduzir uma "família" de sprites através de software, mas eles não serão suficientemente velozes. Um sprite baseado em hardware, no entanto, consegue ter velocidade, pois, como é o caso dos micros da linha MSX, o vídeo dispõe de um rapidíssimo processador, que cuida de todos os detalhes do funcionamento das figuras, sem onerar o processador central.





COMPRESSÃO DE TEXTOS (3)

Agora que você já sabe como comprimir e descomprimir textos, ponha o programa para funcionar e observe seu desempenho no desenvolvimento de um jogo de aventura.



Nos artigos das páginas 1332 e 1404, analisamos diversos algoritmos destinados à compressão de textos. Não explicamos, porém, como utilizar esses programas no contexto de um jogo de aventura. Como você deve se lembrar, nosso objetivo inicial era reduzir o espaço de memória ocupado por uma aventura desse tipo, permitindo a programação de um jogo mais complexo e extenso.

Neste artigo, examinaremos as técnicas de compressão e descompressão de textos aplicadas ao desenvolvimento de um jogo de aventura. Usaremos o algoritmo que demonstrou a maior eficiência de compressão, com a máxima simplicidade de programação: a técnica de compressão de dois códigos por byte, baseada na frequência dos caracteres no texto. Como vimos, esse algoritmo pode ser programado em BASIC sem dificuldades, e é razoavelmente rápido.

DESENVOLVIMENTO DE AVENTURAS

Em uma série de artigos de *Programação de Jogos* (páginas 208, 226, 270, 306 e 394), tratamos das principais técnicas de desenvolvimento de aventuras em BASIC. Com um número mínimo de alterações no programa então fornecido, você poderá empregar as rotinas de compressão e descompressão de textos, economizando uma quantidade razoável de memória.

Se você não armazenou o programa mencionado, digite-o agora, acompanhando as listagens dadas nos sucessivos artigos. Entretanto, cada vez que encontrar uma mensagem tal como:

```
345 PRINT "TIJOLOS SAO MUITO
PESADOS, SEU BRACO DEVE ESTAR
DOENDO."
```

substitua-a por outra mais curta, apenas para lembrar-se do que deve ser definido ou impresso ali. Por exemplo:

```
345 PRINT "TIJOLO PESADO"
```

Procedendo assim, você poderá testar se o programa completo de aventuras está funcionando bem, sem gastar muito tempo digitando mensagens extensas, que depois serão retiradas do programa principal.

Quais são as mensagens e frases que admitem compressão? Em um jogo de aventuras, utilizam-se doze tipos principais de texto alfanumérico:

- nomes de lugares (por exemplo: HALL DE ENTRADA);
- mensagens de localização (VOCE ESTÁ NO HALL DE ENTRADA);
- verbos (NADAR);
- nomes de objetos (REVOLVER);
- mensagens de identificação de objeto (VOCE PEGOU UM REVOLVER);
- mensagens de advertência (ESTÁ MUITO ESCURO PARA VER AS SAÍDAS);
- mensagens de erro (ISTO NÃO PODE SER ESVAZIADO);
- consequências de ações do jogador ou do programa (exemplo: AS BOLINHAS SE ESPALHARAM PELO CHÃO);
- perguntas ao jogador (QUER JOGAR NOVAMENTE?);
- frases comuns, em conjunto com outras mensagens (VOCE PEGOU);
- mensagens de ajuda (VERIFIQUE SE ALGUMA PORTA ESTÁ ABERTA);
- instruções do jogo.

Os jogos de aventuras dificilmente contêm todos esses tipos de texto. Depois de verificar quais deles estão presentes, você deverá decidir se vale a pena comprimi-los. Listas de nomes curtos (objetos e locais, por exemplo) não devem ser comprimidas: o número de bytes economizado possivelmente não compensará os bytes de código de programa gastos para acessar o texto comprimido. Todos os outros tipos de texto (sobretudo as instruções) podem e devem ser comprimidos, desde que tenham duas ou mais palavras: isso levará a um ganho real de espaço.

Para localizar as mensagens que serão comprimidas, o melhor é tirar uma listagem em impressora e marcar todos os pontos em que elas ocorrem. Em seguida, faça uma lista das mensagens, indicando os números das linhas do programa onde elas ocorrem. Agrupe as mensagens segundo o seu tipo: primeiro as mensagens de localização, depois as de identificação de objetos encontrados etc. Deixe as instruções para o fim. Isso facilitará a reprogramação do jogo de aventura e tornará menos cansativa a "caçada" aos erros cometidos.

A LISTA-MESTRE

Tendo todos esses dados, digite no computador um programa em BASIC constituído apenas de linhas **DATA**, com as mensagens que serão comprimidas. Cada mensagem deve ficar em uma linha **DATA** separada.

Preparamos essa listagem para o jogo de aventura publicado em *INPUT*. Ela serve para todos os micros, e começa na linha 5000. Em algumas máquinas é possível digitar o texto em letras minúsculas e a acentuação correta, mas não o faça. Isso é importante, pois, como vimos nos artigos anteriores, empregaremos um código reduzido de representação de caracteres, baseado em sua





É possível comprimir sons e melodias em jogos de aventuras?

Existem técnicas bastante eficientes para a compressão de melodias, mas os algoritmos usados são extremamente especializados. Uma técnica muito simples pode ser empregada, entretanto, pelos usuários dos computadores TRS-Color e MSX: ela consiste em comprimir a seqüência de comandos usados para programar uma melodia como se fosse um texto (variável *string*), e depois descomprimi-la no momento de tocar a melodia.

frequência no texto — diferente, portanto, do ASCII. Como a eficiência de compressão só é obtida se usarmos um máximo de 30-35 caracteres nesse código, trabalharemos apenas com as letras maiúsculas e poucos caracteres de pontuação.



```
5000 DATA "VOCE ESTA DO LADO DE
 FORA DE UM GRANDE PREDIO"
5010 DATA "VOCE ESTA A BEIRA DE
 UM GRANDE RIO"
5020 DATA "VOCE ESTA NUMA FLORE
STA PETRIFICADA"
5030 DATA "VOCE ESTA NUMA SALA
EMPOEIRADA"
5040 DATA "VOCE ESTA NUMA SALA
ESCURA"
5050 DATA "VOCE ESTA EM UM ATAL
HO ENLAMEADO"
5060 DATA "VOCE ESTA NA ENTRADA
 DA CIDADE OCULTA"
5070 DATA "VOCE ESTA NO HALL DE
 ENTRADA"
5080 DATA "VOCE ESTA NO PATIO"
5090 DATA "VOCE ESTA NO JARDIM"
5100 DATA "VOCE ESTA NO GUARDA
LOUCAS"
5110 DATA "VOCE ESTA NA SALA DO
 TRONO"
5120 DATA "ESTA MUITO ESCURO PA
RA VER AS SAIDAS"
5130 DATA "COMO VOCE NAO TEM NA
DA QUE POSSA SER CONFISCADO, EL
E O PRENDE NUMA MASMORRA IMUNDA
"
5140 DATA "DESCULPE, VOCE NAO P
ODE SEGUIR NESSA DIRECAO"
5150 DATA "VOCE NAO PODE LARGAR
 O QUE NAO TEM"
5160 DATA "DESCULPE, NAO POSSO
AJUDAR AGORA"
5170 DATA "TIJOLOS SAO MUITO PE
SADOS, SEU BRACO DEVE ESTAR DOE
NDO"
5180 DATA "JA ESTA ACESA"
5190 DATA "ISTO NAO PODE SER ES
VAZIADO"
5200 DATA "AS BOLINHAS SE ESPAL
HARAM PELO CHAO"
5210 DATA "NADAR AONDE ?"
5220 DATA "QUE VERGONHA, VOCE S
E AFOGOU !"
5230 DATA "VOCE SE MOLHOU TODO"
5240 DATA "VOCE ACHOU UM REVOLV
ER"
5250 DATA "NADA ACONTECE"
5260 DATA "VOCE NAO PODE PUXAR
ISTO"
5270 DATA "VOCE CAIU DENTRO DO
VASO E FOI EMBORA COM A DESCARG
A !"
5280 DATA "PARABENS ! VOCE COMP
LETOU A TAREFA !"
5290 DATA "FIM DO PROGRAMA DE A
VENTURAS"
5300 DATA "QUER JOGAR NOVAMENTE
 (S/N) ?"
5310 DATA "HA UM SACO DE BOLAS
DE GUDE AQUI"
5320 DATA "TEM UM TIJOLO NO CHA
O"
5330 DATA "HA UMA CORRENTE PEND
URADA SOBRE O TRONO"
5340 DATA "TEM UM REVOLVER NO C
HAO"
5350 DATA "UM OLHO CRAVEJADO DE
 BRILHANTES ESTA NO CHAO"
5360 DATA "VOCE ESTA DIANTE DE
UMA LAMPADA"
5370 DATA "DE REPENTE SURGE UM
COLETOR DE IMPOSTOS"
5380 DATA "EU NAO SEI COMO"
5390 DATA "VOCE PEGOU"
5400 DATA "NAO ESTA AQUI"
5405 DATA "PRESSIONE QUALQUER T
ECLA PARA CONTINUAR..."
5410 DATA "INSTRUCOES"
5420 DATA "DEVIDO A UM COLAPSO
FINANCEIRO, VOCE TEVE QUE DEIXA
R O PAIS. SEUS PROBLEMAS VAO TE
RMINAR QUANDO VOCE ENCONTRAR O
LEGENDARIO OLHO CRAVEJADO DE BR
ILHANTES DE UM TOTEM INCA."
5430 DATA "DEPOIS DE TRAZE-LO,
VOCE TERA QUE ENCONTRAR A SAIDA
  CUIDADO COM O COLETOR DE IMPO
STOS !"
5440 DATA "*"
```

Nesse programa, foram codificados estes tipos de texto (com suas linhas):

5000 a 5110: mensagens de localização;
5120 a 5260: mensagens de advertência e erro;
5270 a 5370: consequências de ações;
5380 a 5405: frases comuns e comandos;
5410 a 5440: instruções do jogo.

Observe que só excluímos as listas de objetos, verbos e locais. As linhas com instruções são as mais longas (cuidado para não ultrapassar 255 caracteres), pois isso facilita seu posterior uso.

Depois de digitar o programa, armazene-o em fita ou disco.

CODIFICAÇÃO

A técnica de compressão estatística utiliza um código baseado na lista dos caracteres mais frequentes no texto. Cada código ocupa um nibble (ou seja, quatro bits), e dois nibbles são comprimidos em um byte. Os quinze caracteres mais comuns têm códigos de um nibble (um número de 1 a 15). Os quinze caracteres seguintes, na ordem de freqüência, têm códigos de dois nibbles (um 0, seguido do código de 1 a 15); os próximos quinze têm códigos de três nibbles (dois 0, seguidos de um código de 1 a 15) e assim por diante. Em geral, os quinze caracteres mais frequentes correspondem a 80% ou mais de todo o texto, e é por isso que se obtém uma compressão em torno dos 50%.

Se usarmos uma lista dos caracteres mais frequentes em textos em português, conseguiremos uma boa eficiência de compressão, mas não a ideal. Para alcançar o nível máximo de eficiência, devemos usar a ordenação de caracteres encontrada no próprio texto. Se você quiser determiná-la, siga estas etapas:

- carregue no computador as linhas **DATA** com todas as mensagens;
- acrescente as linhas do programa de contagem de letras fornecido no primeiro artigo da série (página 1332);
- rode o programa e anote o resultado.

Para poupar trabalho ao leitor, preparamos a lista de frequência de caracteres no texto da aventura de *INPUT* (artigo da página 208 e seguintes).

branco	248	H	17
A	176	G	15
E	173	Q	11
O	163	B	10
S	83	F	9
R	79	J	8
D	73	,	7
C	68		6
N	67	'	5
T	64	-	2
U	57	?	2
I	47	X	2
M	47	Z	2
V	42	(	1
L	38	)	1
P	33	/	1

O primeiro conjunto de quinze caracteres (**K$(1)**) é formado por branco, A,E,O,S,R,D,C,N,T,U,I,M,V e L. Os demais caracteres fazem parte, portanto, dos conjuntos seguintes.

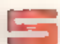
De posse dessa lista, podemos escrever um pequeno programa para testar se a compressão e a descompressão estão funcionando com os textos de exemplo. Carregue novamente na memória somente as linhas **DATA** anteriores, e adicione este programa:



```
10 DIM F$(100),K$(3)
20 K$(1)=" AEOSRDCNTUIMVL":K$(2
)="PHGQBFJ,.!-?XZ(":K$(3)=")/"
180 PRINT "CODIFICANDO..."
190 I=0:K=1
200 READ L$:IF L$="*" THEN GOTO
 230
210 GOSUB 710:I=I+1:F$(I)=X$
215 PRINT I:L$
220 GOTO 200
230 NL=I
240 PRINT:INPUT "NUMERO DA MENS
AGEM ";I
242 IF I=0 THEN GOTO 300
245 IF I<1 OR I>NL THEN PRINT "
*** NAO EXISTE":GOTO 240
250 L$=F$(I):GOSUB 870
290 GOTO 240
300 END
```

Para os computadores das linhas TRS-80 e TRS-Color, adicione a linha:

```
5 CLEAR 2000
```



```
10 DIM f$(100),k$(3,15)
20 LET k$(1)=" AEOSRDCNTUIMVL":
LET k$(2)="PHGQBFJ,.!-?XZ(":LET
 k$(3)=")/"
180 PRINT "CODIFICANDO..."
190 LET i=0:LET k=1
200 READ L$:IF L$="*" THEN GOTO
 230
210 GOSUB 710:LET i=i+1:LET f$(
i)=x$
215 PRINT i;L$
220 GOTO 200
230 LET nL=I
240 PRINT:INPUT "NUMERO DA MENS
AGEM ";i
242 IF i=0 THEN GOTO 300
245 IF i<1 OR i>nL THEN PRINT "
*** NAO EXISTE":GOTO 240
250 LET L$=f$(i):GOSUB 870
290 GOTO 240
300 STOP
```

As sub-rotinas que começam nas linhas 710 (codificação) e 870 (decodificação) podem ser copiadas integralmente do programa listado no primeiro artigo, para os computadores respectivos.

A linha 20 do programa define os três conjuntos de códigos de caracteres. O laço formado pelas linhas 180 a 220 comprime o texto, lendo o linha por linha em **DATA** e colocando o resultado no conjunto **F$** (uma mensagem por linha). A sub-rotina de codificação é chamada em 210.

Finalmente, as linhas 240 a 290 solicitam ao usuário o número da mensagem que deseja imprimir. Digite 0, se quiser interromper o programa.

ARMAZENAGEM DO RESULTADO



Vamos agora desenvolver dois programas separadamente. O primeiro funciona como o programa anterior, servindo para comprimir todas as mensagens e instruções. O resultado, gravado em fita ou disco, em um arquivo seqüencial, será depois carregado pelo programa de jogo propriamente dito. Assim, a rotina que executa a compressão não precisa ser incluída no programa do jogo. Neste se introduzem apenas as rotinas responsáveis pelo carregamento, descompressão e impressão.

O segundo programa compõe-se de duas rotinas que podem ser acrescentadas ao jogo de aventura. Uma delas lê o arquivo seqüencial com o texto comprimido; a outra efetua a descompressão quando solicitado. Explicaremos mais adiante como usá-las.

No programa-exemplo, o texto comprimido foi armazenado em um conjunto **F$**. Portanto, bastaria escrever uma sub-rotina de gravação desse conjunto (com texto comprimido) e, depois, uma sub-rotina para lê-lo, de volta à memória, no programa de jogo.

Parece tudo muito simples, mas há um senão: trabalhando com conjuntos alfanuméricos, como o **F$**, no programa que utilizará o texto comprimido, ocuparemos um espaço de memória, que ultrapassará, em tamanho, o exigido pelo texto original, descomprimido.

Devido ao modo como o BASIC trata cadeias de caracteres (*string*), é difícil evitar esse problema. Usaremos, assim, um conjunto numérico **T%** para armazenar os bytes comprimidos. Os dois bytes gerados pela compressão de cada quatro nibbles serão armazenados em um conjunto de **T%**, pois uma variável inteira (em todos os computadores, exceto o Spectrum) ocupa dois bytes, e pode armazenar um número entre −32768 e 32767. O tamanho de **T%** será igual ao número de caracteres comprimidos (no exemplo anterior, 447 bytes).

Para localizar o início de cada mensagem nessa seqüência contínua de códigos, temos que criar um segundo con-

MICRO DICAS

ACENTUANDO TEXTOS COMPRIMIDOS

Para os perfeccionistas da programação, é frustrante não poder usar caracteres acentuados em um texto que será comprimido pelo algoritmo estatístico discutido neste artigo. Mas, em geral, a falta de acentos não dificulta a leitura do texto.

Entre as poucas exceções destaca-se a palavra *é*, que, sem acento, pode impedir a compreensão do texto. Para evitar esse problema sem aumentar o número de caracteres do conjunto utilizado, recorra ao apóstrofo coloque-o logo após a letra que deve ter acento agudo, sem deixar espaço o resultado será satisfatório.

junto A%, que conterá os apontadores, ou índices de T%. O comprimento de A% será igual ao número de mensagens mais 1 (no nosso exemplo, 46).

Se dimensionamos T% e A% em seu limite, obteremos o máximo de economia de espaço de memória.

A rotina de codificação é a seguinte (o programa de teste vem depois):



```
700 REM - ROTINA DE CODIFICACAO
710 N=0·LN=0
720 FOR J=1 TO LEN(L$)
730 C$=MID$(L$,J,1)
740 P=INSTR(K$(K),C$)
750 N=N+1:C%(N)=P·LN=N
755 IF N<4 THEN GOTO 775
760 T%(NC)=(C%(1) OR (16*C%(2)
))+256*(C%(3) OR (16*C%(4)))-
32768
765 NC=NC+1:N=0
775 IF P=0 THEN K=K+1:GOTO 740
776 K=1
780 NEXT J
785 IF LN=4 THEN RETURN
790 FOR J=LN+1 TO 4:C%(J)=1
792 NEXT J
795 T%(NC)=(C%(1) OR (16*C%(2)
))+256*(C%(3) OR (16*C%(4)))-
32768
796 NC=NC+1:RETURN
```



Para executar a rotina nos micros compatíveis com o Apple, acrescente:

```
740 FOR P=1 TO 15:IF C$=MID$(K$
(K),P,1) THEN GOTO 750
745 NEXT P·P=0
```

A sub-rotina é igual à apresentada no artigo anterior, com uma diferença: os nibbles são armazenados de quatro em quatro, no conjunto C%, e comprimidos pelas linhas 760 e 795. O valor 32768 é diminuído do byte assim comprimido, para que seu conteúdo fique entre −32768 e 32767.

Segue-se o programa principal:



```
10 DIM K$(3),T%(450),A%(46)
15 DIM N%(2),C%(4)
20 K$(1)=" AEOSRDCNTUIMVL":K$(2
2)="PHGQBFJ,.!-?XZ("·K$(3)=")/"
180 PRINT "CODIFICANDO..."
190 I=0:K=1·NC=1
200 READ L$:IF L$="*" THEN GOTO
 230
205 I=I+1:A%(I)=NC
210 GOSUB 710
215 PRINT I;A%(I);L$
220 GOTO 200
230 NL=I:A%(I+1)=NC
300 INPUT "GRAVAR (S/N) ";R$
310 IF R$="S" THEN GOSUB 400
320 END
```

Para os computadores das linhas

TRS-80 e TRS-Color, acrescente a linha:

```
5 CLEAR 2000
```

A sub-rotina para o armazenamento em arquivo sequencial começa na linha 400, e grava os conjuntos **T** (com os códigos comprimidos) e **A** (apontadores):

```
400 REM - ROTINA DE GRAVACAO
410 INPUT "NOME DO ARQUIVO";N$
420 OPEN "O",#-1,N$
430 PRINT# 1,NL,NC
440 FOR I=1 TO NL+1:PRINT#-1,A$
(I):NEXT I
450 FOR I=1 TO NC:PRINT# 1,T$(I
):NEXT I
460 CLOSE#-1:RETURN
```

```
400 REM - ROTINA DE GRAVACAO
410 INPUT "GRAVADOR PRONTO ",N$
430 PRINT#-1,NL,NC
440 FOR I=1 TO NL+1:PRINT#-1,A$
(I):NEXT I
450 FOR I=1 TO NC:PRINT#-1,T$(I
):NEXT I
460 RETURN
```

```
400 REM - ROTINA DE GRAVACAO
410 INPUT "NOME DO ARQUIVO";N$
420 OPEN "CAS:"+N$ FOR OUTPUT
AS #1
430 PRINT#1,NL,NC
440 FOR I=1 TO NL+1:PRINT#1,A$
(I):NEXT I
450 FOR I=1 TO NC:PRINT#1,T$(I
):NEXT I
460 CLOSE#1:RETURN
```



```
400 REM - ROTINA DE GRAVACAO
410 N%(1)=NL:N%(2)=NC
420 INPUT "GRAVADOR PRONTO ",N$
430 STORE N%:STORE A%:STORE T%
460 RETURN
```

Para executar o programa completo de compressão, acrescente ao final as linhas **DATA**. O computador executará toda a tarefa automaticamente. À medida que codifica as linhas, o programa as exibe na tela, perguntando se o usuário deseja gravar o resultado. Se a resposta for **S**, os conjuntos **A** e **T** serão armazenados para uso posterior pela rotina de decodificação.

No Spectrum não há tratamento separado para arquivos sequenciais em fita, a não ser que se disponha de um Microdrive (não existente no Brasil). Por isso, o programa para essa máquina apresenta uma abordagem diferente dos demais micros.

Inicialmente, carregue o jogo de aventura completo e acrescente as linhas abaixo (renumere-as para compatibilizar com o programa do jogo, se isto se fizer necessário).

```
10 DIM k$(3,15),t(450),a(46)
15 DIM c(4)
20 LET k$(1)=" AEOSRDCNTUIMVL":
LET k$(2)="PHGQBFJ,.!-?XZ(":LET
 k$(3)=")/"
190 LET i=0:LET k=1:LET nc=1
200 PRINT "ENTRE A MENSAGEM ";:
LINE INPUT L$: IF L$="*" THEN GO
TO 230
205 LET i=i+1:LET a(i)=nc
210 GOSUB 710
215 PRINT i,a(i),L$
220 GOTO 200
230 LET nl=i:LET a(i+1)=nc
240 STOP
```

O programa principal pedirá ao usuário para entrar as mensagens pelo teclado, uma de cada vez. Quando chegar à última, responda com um asterisco, para assinalar o fim.

Usaremos um conjunto numérico **T** para armazenar os bytes comprimidos. Os dois bytes gerados pela compressão de cada quatro nibbles serão armazenados em um elemento de **T**. O tamanho desse conjunto será igual ao número de caracteres comprimidos (no nosso exemplo, 447 bytes). Para localizar o início de cada mensagem nessa seqüência contínua de códigos, temos que criar um segundo conjunto, **A**, que conterá os apontadores, ou índices de **T**. O comprimento de **A** será igual ao número de mensagens mais 1 (no exemplo, 46).

Se dimensionarmos **T** e **A** no limite, obteremos o máximo de economia de memória. Acrescente a rotina de codificação:

```
700 REM - ROTINA DE CODIFICACAO
710 LET n=0:LET Ln=0
```

```
720 FOR j=1 TO LEN L$
730 LET c$=L$(j)
740 FOR p=1 TO 15:IF c$=k$(k,p)
 THEN GOTO 750
745 NEXT p·LET p=0
750 LET n=n+1·LET c(n)=P·LET L
n=n
755 IF n<4 THEN GOTO 775
760 LET t(nc)=(c(1) OR (16*c
(2)))+256*(c(3) OR (16*c(4)))
 32768
765 LET nc=nc+1·LET n=0
775 IF p=0 THEN LET k=k+1:GOTO
740
776 LET k=1
780 NEXT j
785 IF Ln=4 THEN RETURN
790 FOR j=Ln+1 TO 4·c(j)=1·NEX
T j
795 LET t(nc)=(c(1) OR (16*c
(2)))+256*(c(3) OR (16*c(4)))
-32768
796 LET nc=nc+1 RETURN
```

A sub-rotina é igual à apresentada no artigo anterior, com uma diferença: os nibbles são armazenados de quatro em quatro, no conjunto C%, e comprimidos pelas linhas 760 ou 795. O valor 32768 é diminuído do byte assim comprimido, para que seu conteúdo fique entre −32768 e 32767.

Depois de acrescentar o programa principal e a rotina de decodificação ao jogo da aventura, execute-o. Após entrar todas as mensagens, grave o programa e os dados juntos, em fita.

DESCOMPRESSÃO

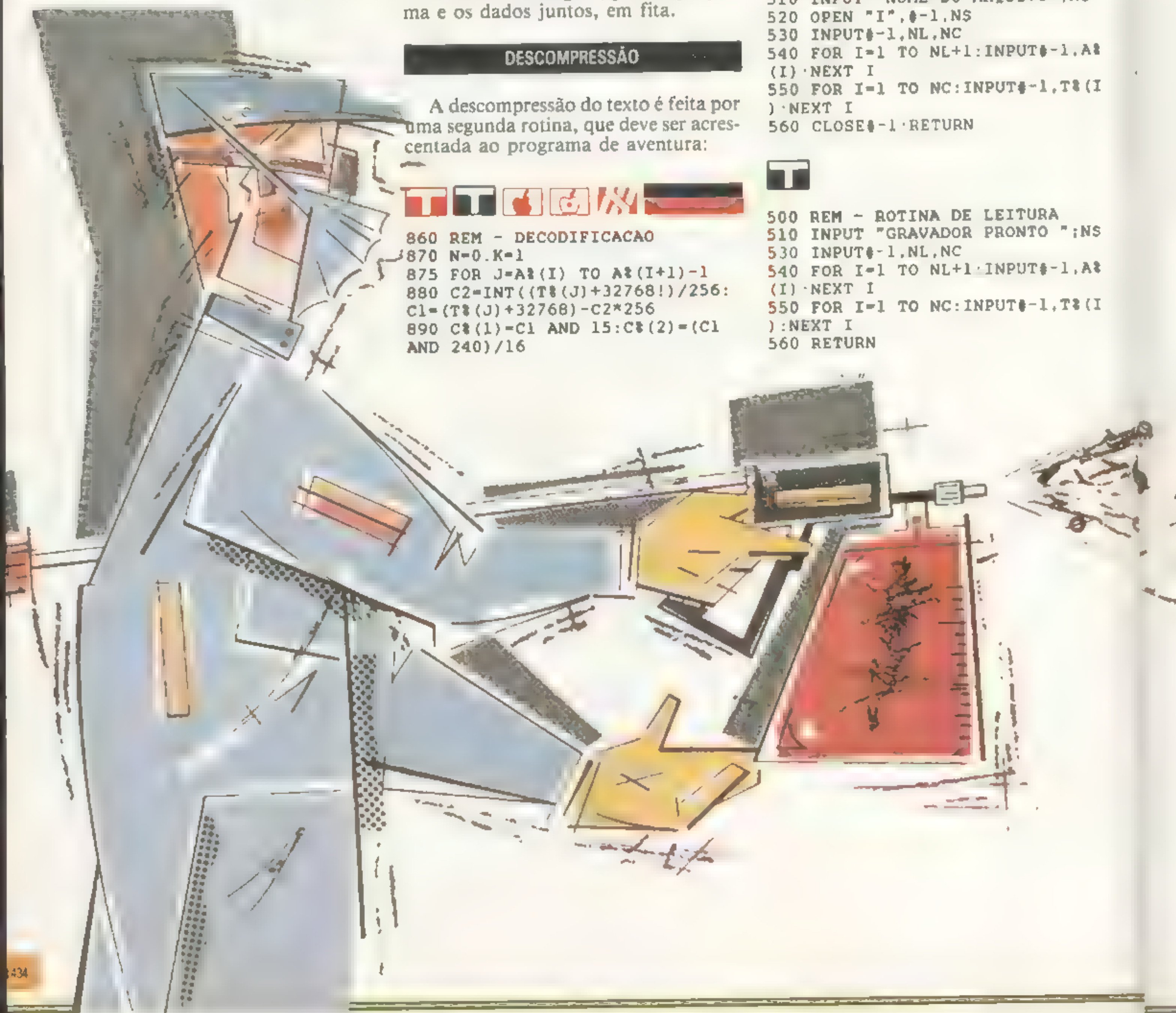
A descompressão do texto é feita por uma segunda rotina, que deve ser acrescentada ao programa de aventura:

```
860 REM - DECODIFICACAO
870 N=0.K=1
875 FOR J=A%(I) TO A%(I+1)-1
880 C2=INT((T%(J)+32768!)/256:
C1=(T%(J)+32768)-C2*256
890 C%(1)=C1 AND 15:C%(2)=(C1
AND 240)/16
895 C%(3)=C2 AND 15:C%(4)=(C2
AND 240)/16
900 FOR L=1 TO 4
910 IF C%(L)=0 THEN K=K+1:GOTO
 930
920 PRINT MID$(K$(K),C%(L),1);:
K=1
930 NEXT L:NEXT J
950 PRINT:RETURN
```

Complete o programa com a rotina de carregamento dos conjuntos **A** e **T**, com os dados no arquivo sequencial:

```
500 REM - ROTINA DE LEITURA
510 INPUT "NOME DO ARQUIVO";N$
520 OPEN "I",#-1,N$
530 INPUT#-1,NL,NC
540 FOR I=1 TO NL+1:INPUT#-1,A%
(I)·NEXT I
550 FOR I=1 TO NC:INPUT#-1,T%(I
)·NEXT I
560 CLOSE#-1·RETURN
```

```
500 REM - ROTINA DE LEITURA
510 INPUT "GRAVADOR PRONTO ";N$
530 INPUT#-1,NL,NC
540 FOR I=1 TO NL+1·INPUT#-1,A%
(I)·NEXT I
550 FOR I=1 TO NC:INPUT#-1,T%(I
):NEXT I
560 RETURN
```





```
500 REM - ROTINA DE LEITURA
510 INPUT "NOME DO ARQUIVO";N$
520 OPEN "CAS:"+N$ FOR INPUT AS
#1
530 INPUT#1,NL,NC
540 FOR I=1 TO NL+1:INPUT#1,A%
(I):NEXT I
550 FOR I=1 TO NC:INPUT#1,T%(I
) NEXT I
560 CLOSE#1:RETURN
```



```
500 REM - ROTINA DE LEITURA
510 INPUT "GRAVADOR PRONTO ";N$
530 RECALL N%.RECALL A%:RECALL
T%.NL=N%(1):NC=N%(2)
560 RETURN
```

Apresentamos a seguir um pequeno programa de teste. Com ele, você terá oportunidade de verificar o funcionamento conjunto das rotinas de leitura e de descompressão:



```
10 DIM K$(3),T%(450),A%(46)
20 K$(1)=" AEOSRDCNTUIMVL":K$(2
)="PHGQBFJ,.!-?XZ(":K$(3)=")/"
30 GOSUB 510
240 PRINT:INPUT "NUMERO DA MENS
AGEM ";I
242 IF I=0 THEN GOTO 300
245 IF I<1 OR I>NL THEN PRINT "
*** NAO EXISTE":GOTO 240
250 GOSUB 870
290 GOTO 240
300 END
```

As linhas 10, 20 e 30 desse programa devem ser colocadas no início do programa do jogo. Elas dimensionam os conjuntos de trabalho do descompressor de textos e chamam a rotina 510. Esta carrega os conjuntos **T** e **A**, a partir do texto comprimido que está armazenado na memória auxiliar.

As linhas 240 a 300 simplesmente imprimem a mensagem solicitada pelo usuário e mostram como a sub-rotina 870 (de decodificação) deve ser usada dentro do programa de aventura.

COMO ADAPTAR O PROGRAMA

Carregue de novo o programa de aventura e localize as linhas que contêm mensagens incluídas na lista de compressão. Suponhamos que você encontre a seguinte linha (abreviada, como recomendamos no começo deste artigo)

```
345 PRINT "TIJOLO PESADO"
```

Substitua-a por:

```
345 LET I=17:GOSUB 870
```

Essa linha iguala o apontador **I** ao número da mensagem na lista comprimida, e passa o controle para a sub-rotina de descompressão. Esta localiza o início e o fim da mensagem, no conjunto de apontadores **A**, e imprime na tela o texto descomprimido.

Repita esse procedimento com todas as linhas onde ocorrem mensagens, e grave o resultado final. Não se esqueça de adicionar as linhas 10, 20 e 30 do nosso exemplo (renumeradas, se for o caso).



Para o Spectrum, o esquema é um pouco diferente. Carregue o programa de aventura que gravou em fita, juntamente com o programa de compressão e o texto comprimido. Apague as linhas 190 a 796 e modifique cada uma das linhas com mensagens, como foi explicado antes. Em seguida, acrescente:

```
860 REM - DECODIFICACAO
870 LET n=0 LET k=1
875 FOR j=a(i) TO a(i+1)-1
880 LET c2=INT((t(j)+32768)/256
:LET c1=(t(j)+32768) c2*256
890 LET c(1)=c1 AND 15 LET c(2)
=(c1 AND 240)/16
895 LET c(3)=c2 AND 15:LET c(4)
=(c2 AND 240)/16
900 FOR L=1 TO 4
910 IF c(L)=0 THEN LET k=k+1:G
OTO 930
920 PRINT K$(K,c(L)), LET k=1
930 NEXT L NEXT j
950 PRINT RETURN
```

Armazene a nova versão em fita. Para rodar o programa completo, use um **GOTO** para a primeira linha do programa de aventura (não a linha 10 do programa de codificação). Nunca empregue o comando **RUN**, pois você perderá todo o texto das mensagens.

PROGRAMAÇÃO EM PASCAL

Muitos consideram o Pascal como a melhor alternativa para o BASIC. Com seus poderosos recursos, podem-se elaborar programas bem estruturados e fáceis de documentar e testar.

Para quem esta familiarizado com apenas uma linguagem de programaçao, o aprendizado de outra pode apresentar maior ou menor dificuldade - o que depende, sobretudo, do grau de semelhança entre as duas linguagens.

Se a estrutura e sintaxe de ambas forem parecidas, basta fazer uma tradução o novo "vocabulario" e aprendido, e as regras de formação de instruções são aplicadas da mesma maneira. As vezes, porém, a diferença entre as linguagens é tão grande, que o aprendiz se vê forçado a abrir mão das regras que conhece e adotar um outro raciocínio.

A linguagem LOGO, por exemplo, examinada em artigos anteriores, foi projetada para facilitar ao maximo a introdução de iniciantes ao mundo da programação. Entretanto, para quem foi "criado" em BASIC — como a maioria dos usuários de micros pessoais —, a adaptação pode apresentar problemas. Algumas vezes, chega a ser difícil aceitar as afirmativas de que LOGO foi feito para principiantes!

Isso ocorre, fundamentalmente, porque o LOGO tem uma estrutura interna bastante diferente da do BASIC, embora seu vocabulário seja mais simples. LOGO é uma linguagem mais bem estruturada, e trabalha com listas hierárquicas, como o LISP. Essas características estimulam o programador iniciante a adotar padrões estruturados de raciocínio e de resolução de problemas.

A programação estruturada também é um aspecto importante de uma linguagem imperativa, o Pascal. Como vimos no artigo da página 1288, ele faz parte da família de linguagens algorítmicas e declarativas iniciadas com o ALGOL. Sua filosofia pode parecer totalmente estranha ao programador BASIC. Ainda que experiente, este está acostumado a elaborar programas no teclado, testando e alterando seções à medida que progride, até chegar a um resultado que, muitas vezes, evidencia um objetivo não muito claro no começo do trabalho. O Pascal so lhe parecera menos complicado se ele tiver desenvolvido hábitos de programação estruturada — em BASIC ou outra linguagem.

Por que então dar-se ao trabalho de aprender Pascal? Além de permitir a elaboração de programas mais bem estruturados e de simplificar a documentação e a execução de testes, essa linguagem apresenta certas vantagens sobre o BASIC. Em programas muito complexos, o Pascal produz um codigo mais claro e curto. A facilidade em definir procedimentos possibilita a programação modular e a formação de bibliotecas de procedimentos, que podem ser aproveitados integralmente em outros programas. Finalmente, as versões modernas do Pascal são adequadas tanto para programação científica (cálculos matematicos) quanto para o desenvolvimento de aplicações comerciais.



FUNDAMENTOS DO PASCAL

O Pascal foi criado em 1970 pelo professor Niklaus Wirth, do Instituto Federal de Tecnologia em Zurique, Suíça. Seu nome é uma homenagem a Blaise Pascal, filósofo e matemático do século XVII, responsável por importantes descobertas científicas e inventor de uma das primeiras máquinas de calcular mecânicas, a *Pascalina*.

O objetivo de Wirth era desenvolver uma linguagem destinada a ensinar os conceitos fundamentais de estruturas de programação. Essa linguagem deveria estimular o estudante a elaborar a estrutura do programa antes de começar a escrevê-lo. Nela Wirth introduziu uma série de recursos a fim de permitir que a solução programada contenha a própria informação a ser processada.

Em outras palavras, para se escrever um programa em Pascal é necessário planejar como resolver o problema. A solução escolhida e refinada sucessivamente, cobrindo detalhes cada vez menores, até se alcançar o nível de um procedimento (unidade algorítmica). Só se trabalha no programa principal depois que todos os procedimentos necessarios estiverem programados.

Para desenvolver o Pascal, Wirth utilizou muitas das idéias em que se baseia o ALGOL 606. Originalmente orientado para o ensino de linguagens estruturadas em computadores de grande porte, o PASCAL foi adaptado para uso em microcomputadores. Hoje, é extensamente empregado tanto no ensino quanto em aplicações comerciais em micros.

PASCAL PARA MICROS

Existem versões do Pascal para diversas linhas de micros pessoais e profissionais. As versões reduzidas (que não têm todos os comandos do Pascal padronizado), chamadas Tiny-Pascal, podem ser carregadas de fita cassete. Porém, o Pascal é mais eficiente quando usado com sistemas baseados em disco, como o UCSD Pascal, para o Apple.

Como são muitas as versões comerciais dessa linguagem, não forneceremos

listagens específicas para cada linha de microcomputador — como fizemos para o BASIC e o LOGO. Mas, mesmo que não tenha um compilador Pascal, não se preocupe: você poderá entender os elementos básicos da linguagem, sem precisar rodar o programa em um computador. Afinal, o desenvolvimento de programas longe da máquina é uma das principais características do Pascal.

Existe outra diferença fundamental entre o BASIC usado em microcomputadores pessoais e o Pascal: ele é uma linguagem *compilada*.

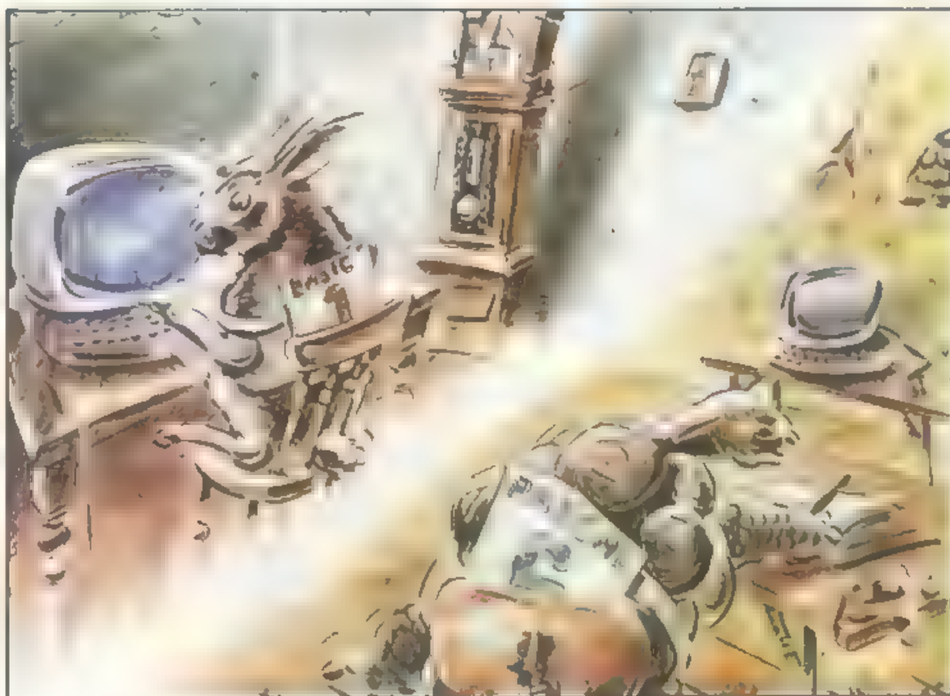
O BASIC adotado na maioria dos micros é uma linguagem *interpretada*: cada vez que o interpretador encontra uma instrução em BASIC, converte-a para códigos de máquina e a executa. Assim, para um laço simples como:

```
10 FOR N=1 TO 100
20 PRINT N
30 NEXT N
```

cada declaração nas linhas 10 a 30 será traduzida cem vezes para código de máquina! Não surpreende, portanto, que o BASIC seja considerado uma linguagem lenta. Outra desvantagem é que o interpretador precisa permanecer na memória o tempo todo, ocupando um espaço valioso, que poderia ser utilizado pelo programa ou dados do usuário.

Uma linguagem compilada funciona de maneira diferente. Uma vez que se tenha entrado o programa na máquina, na forma de um texto (o *programa-fonte*), ele é traduzido de uma vez só para código de máquina (o *programa-objeto*), e é este que o computador executa. Depois de feita a tradução, o programa compilador não é mais necessário, ficando armazenado em fita ou disco. Em conseqüência, mais espaço na memória é liberado para uso do programa aplicativo desenvolvido pelo usuário.

As linguagens compiladas são muito mais rápidas que as interpretadas, pois o demorado processo de tradução para código de máquina é executado apenas uma vez. Embora existam compiladores para o BASIC, essa linguagem não foi projetada para ser compilada; assim, seus programas-objeto são longos e, freqüentemente, tão lentos quanto o programa-fonte. O Pascal, ao contrário, foi projetado especificamente para ser compilado, permitindo a obtenção de programas tão compactos e rápidos quanto os escritos diretamente em Assembler. E há uma vantagem adicional: é mais simples desenvolver um programa em Pascal do que em Assembler.



O Pascal apresenta, portanto, um equilíbrio bem satisfatório entre velocidade de desenvolvimento e de execução. Somente o C, uma linguagem semelhante a ele em estrutura e filosofia, apresenta maior eficiência.

PROJETO ALGORÍTMICO

Como vimos, a programação em Pascal requer uma fase inicial de projeto, antes de se chegar à máquina. Nessa fase, utilizam-se certas ferramentas de planejamento que ainda não discutimos em *INPUT*, entre as quais o fluxograma. Este deve ser usado com cautela, pois, em si, não é estruturado. Uma ferramenta mais conveniente para o projeto de programas é o *algoritmo*: um detalhamento passo a passo do método a ser empregado na solução do problema.

Um exemplo familiar de um algoritmo é a receita culinária: ela contém toda a informação necessária para reproduzir um certo prato. Um programa finalizado tem, igualmente, toda a informação de que o computador precisa para resolver um problema ou realizar uma tarefa de um determinado modo.

Entretanto, o programador (ou o cozinheiro), quando começa a escrever um programa (ou a receita), não tem uma idéia bem definida sobre o procedimento a seguir para chegar ao objetivo desejado. Como exemplo, vamos analisar o processo de elaboração de uma receita de torta. Em uma primeira tentativa, poderíamos dividir a tarefa (ou algoritmo inicial) em quatro passos:

1. Preparar a massa
2. Preparar o recheio
3. Colocar tudo numa fôrma
4. Assar a torta

Ao fazer uma torta, talvez você não precise cumprir todos os passos (é pos-

sível comprar a massa pronta, por exemplo!). Suponhamos, porém, que o computador é um "cozinheiro amador", que deve conhecer cada detalhe.

A próxima etapa consiste em fazer a subdivisão dos passos em operações menores — processo denominado *refinamento gradativo*. Você encontrará esse conceito na metodologia de desenvolvimento de programas muito extensos, que não podem ser projetados de uma só vez. No nosso exemplo, refinando o passo 1, teríamos:

1.1 - Pesar a farinha
1.2 - Pesar a manteiga
1.3 - Misturar a manteiga com a farinha
1.4 - Adicionar água
1.5 - Mexer

Poderíamos refinar ainda mais cada passo, até chegarmos a um nível de simplicidade tal que fosse possível a qualquer um fazer uma torta.

O principal, nesse processo, foi termos considerado previamente todos os passos da solução do problema, em vez de iniciarmos a tarefa em um ponto qualquer, ao acaso. Para a programação em Pascal, o método a ser seguido é muito similar. No estágio inicial, escreve-se cada passo em português mesmo. Refinando-os sucessivamente, chega-se à formalização proporcionada por uma declaração ou instrução em Pascal. Separados do algoritmo inicial, os passos podem ser representados por procedimentos ou funções. Estes são posteriormente reunidos em um programa só. Mas como escrever um programa em Pascal sem conhecer os comandos dessa linguagem? Examinemos alguns exemplos.

UM PROGRAMA EM PASCAL

Os programas em Pascal costumam parecer complicados demais em relação à simplicidade da tarefa a cumprir. Veja, por exemplo, este programa, que soma dois números e exibe o resultado:

```
program exemplo (input,output);
var n1,n2,resultado integer;
begin
   read (n1,n2);
   resultado :=n1+n2;
   writeln (resultado)
end
```

Como se pode notar, o programa é mais longo que um equivalente em BASIC, embora de execução mais rápida. Quanto maior a complexidade da tarefa, porém, maiores a economia e concisão oferecidas por um programa em Pascal.

O programa anterior poderia resultar de um algoritmo inicial como:

1. Estabeleça as condições iniciais
2. Inicie o procedimento
3. Entre os números
4. Acrescente os números
5. Imprima o resultado
6. Termine o procedimento

O programa segue rigorosamente o algoritmo. Observe que é necessário declarar explicitamente os passos 1, 2 e 6 — ou seja, precisamos especificar o que é trabalhado pelo programa, quando iniciá-lo e quando encerrá-lo.

Este é um dos fatores que explicam a maior extensão de um programa escrito em Pascal em relação a um equivalente em BASIC. Vamos agora analisar o programa linha por linha:

```
program exemplo (input,output);
```

significa que o programa terá entradas (**input**) e saídas (**output**) e se chamará **exemplo**. Se ele não precisasse ler dados externos, bastaria escrever:

```
program exemplo (output);
```

O ponto e vírgula no final de uma linha indica o fim de uma declaração.

As características de todas as variáveis usadas por um programa em Pascal devem ser especificadas. A linha:

```
var n1,n2,resultado:integer;
```

informa que usaremos três variáveis inteiras (**integer**). **n1**, **n2** e **resultado**.

Feitas as declarações, assinalamos o início do procedimento propriamente dito, com a palavra begin. Note agora que o texto do programa é recuado em relação à linha anterior. Este é um recurso muito usado em programas estruturados, pois indica para quem lê quais são as declarações dentro de um procedimento ou laço de repetição.

A leitura dos dados pelo teclado é muito semelhante à do BASIC: read (n1,n2). Só podemos usar aqui esses nomes de variáveis porque elas foram declaradas antes, em var.

A declaração aritmética também é semelhante a uma instrução em BASIC:

```
resultado :=n1+n2;
```

O símbolo de atribuição := foi assim definido para evitar a confusão causada pelo = do FORTRAN e do BASIC, que dá a impressão errônea de se tratar de uma fórmula matemática.

Finalmente, escrevemos o resultado:

```
writeln (resultado)
```

que tem um efeito semelhante ao **PRINT RESULTADO** de um programa em BASIC. Não há ponto e vírgula no final da linha, pois a declaração seguinte é um **end**. Alguns compiladores permitem o uso desse sinal desde que se coloque uma linha em branco depois.

O Pascal conta, naturalmente, com vários outros comandos e funções, admitindo também a introdução de notas e comentários no programa, entre chaves: e . Com os microcomputadores que não têm essas teclas, pode-se utilizar uma combinação de parênteses com asteriscos (* e *).

EDIÇÃO DO PROGRAMA

O processo de edição e teste de programas em linguagem compilada, como o Pascal, não é tão direto como com um interpretador. Normalmente, edita-se



apenas o programa-fonte, com o auxílio de um editor de linhas (quase sempre disponível como um utilitário do sistema operacional) ou de um processador de textos. Depois, compila-se o programa e testa-se o resultado, executando-o. Caso se verifique algum erro, todo o processo deve ser repetido, a partir do programa-fonte.

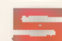
Alguns compiladores Pascal, como o UCSD e o Turbo-Pascal, dispõem de um editor próprio, o que evita esse vai vém entre editor, sistema operacional e compilador. Nesse caso, tanto o programa-fonte quanto o programa-objeto podem residir na RAM enquanto durar o processo, agilizando-o muito — se o programa não for muito longo, e se houver boa capacidade de memória.

Como está em linguagem de máquina pura, sem comentários, rótulos, ou similares, o programa compilado oferece a vantagem final de inviabilizar a ação de "espiões". Assim, depois de editar um programa em Pascal, você poderá ter a certeza de que ninguém copiará os algoritmos de sua criação.

PROGRAMAS DISPONÍVEIS

Existem compiladores Pascal para a maioria das linhas de microcomputadores. Entretanto, como já mencionamos, eles variam muito entre si, quanto ao funcionamento e desempenho, e nem todos podem ser encontrados com facilidade no Brasil. Antes de adquirir um programa desse tipo, convém fazer uma pesquisa em diferentes "software houses" — de preferência nas especializadas em uma determinada linha de computadores ou nas dedicadas exclusivamente à programação Pascal.

O uso mais sério do Pascal requer componentes mais caros, exigindo, no mínimo, um acionador de disquetes. Por isso, máquinas como o Apple e a IBM-PC contam com as melhores versões.



O compilador Pascal para o Spectrum é exclusivo para microdrives e disquetes (não disponíveis no Brasil). Inclui um editor de linhas, um gerador de números aleatórios e suporte gráfico em alta resolução.



Existem diversas versões do Pascal para o TRS-Color. Suas características variam conforme o sistema operacional utilizado: Flex, RS-DOS (original) ou OS-9 (semelhante ao Unix). Todas funcionam somente em micros dotados de acionadores de disquetes.



Esse micro dispõe de duas versões do Pascal: um compilador que roda sob o sistema operacional TRSDOS e compatíveis, e outro, semelhante ao Turbo Pascal, que roda sob o sistema operacional CP/M. Ambas são bastante completas, mas não incluem suporte gráfico.



As duas versões mais utilizadas de Pascal para as máquinas compatíveis com a linha Apple são o UCSD e o Turbo-Pascal. O UCSD (abreviatura da universidade onde foi desenvolvida, *University of California at San Diego*) foi o primeiro e mais popular *sistema* Pascal (*p-System*). Trata-se, na realidade, de um sistema operacional completo, que substitui o DOS, e não apenas de um compilador.

Já o Turbo-Pascal roda sob o sistema operacional CP/M (*Control Program for Microcomputers*), que pode ser colocado em um Apple quando se troca a UCP por um microprocessador Z-80.



O Pascal para o MSX é do tipo Microsoft e destina-se a máquinas dotadas de acionador de disquetes de 5.25 ou 3.5 polegadas, rodando sob o sistema operacional MSX-DOS ou compatível.

FORMATAÇÃO DE VALORES

- O PRINT USING
- VALORES MONETÁRIOS
- FORMATAÇÃO COM PARÊNTESES
- UMA ROTINA PODEROSA

Se você tem problemas com a formatação de saída para valores monetários, poderá encontrar aqui uma boa solução: o uso de duas eficientes rotinas no lugar do comando PRINT USING.

O comando **PRINT USING** é um dos recursos mais poderosos do BASIC de algumas linhas de computadores, como o TRS-80, o TRS-Color e o MSX. Com ele, podemos formatar de diferentes maneiras a saída de números e cadeias alfanuméricas. Porém, esse comando apresenta inconvenientes, sobretudo quanto a formatação de valores monetários.

No Brasil, usam-se variáveis de precisão dupla (quinze dígitos) para representar valores monetários, pois a precisão simples não nos permite lidar com valores superiores a 10 milhões de cruzados. E o **PRINT USING**, infelizmente, é muito lento quando se trata de formatar variáveis de precisão dupla.

Além disso, não é possível utilizar o **PRINT USING** dos computadores mencionados para formatar valores monetários segundo a convenção brasileira, que impõe a separação dos milhares por pontos e dos centavos por vírgulas.

As funções que explicamos neste artigo oferecem várias alternativas para a formatação de valores, e são muito mais rápidas (três a seis vezes) do que o **PRINT USING** normal.

FORMATAÇÃO COM PARÊNTESES



Esta rotina permite a formatação de valores monetários segundo as convenções dos balanços contábeis: com os números negativos entre parênteses:

```
10 DEF FNNS(V#,E%)=RIGHTS(STRIN
GS(E%," ")+LEFTS("(",ABS(V#<0))
+LEFTS(" ",ABS(V#>=0))+MIDS(STR
$(V#),2),E%)+LEFTS(")",ABS(V#<0
))+LEFTS(" ",ABS(V#=>0))
100 INPUT "Numero de colunas ";
E%
110 INPUT "Valor ";V#
120 PRINT FNNS(V#,E%) GOTO 110
```

A função **FN$**, estabelecida na linha 10, tem dois argumentos: **V #**, um valor monetário em precisão dupla (pode incluir centavos ou não), e **E%**, um valor inteiro que especifica o número de colunas a usar na formatação (com justificação à direita). As linhas 100 e 120 formam um laço simples de teste.

O BASIC das máquinas mencionadas é capaz de definir em apenas uma linha expressões de grande complexidade — inclusive expressões condicionais que imitam a operação de um **IF...THEN**. Observe, por exemplo, a expressão:

```
LEFTS("(",ABS(V#<0))
```

Pode parecer estranho utilizar um operador de comparação (sinal de menor) dentro de uma expressão matemática. Mas podemos compreender sua função se levarmos em conta que o resultado numérico de uma expressão lógica é igual a −1 (ou 1, em alguns computadores), se ela for verdadeira, e a 0, se for falsa. Assim, se **V #** for menor que 0, a expressão anterior resultará em um *string* contendo um parêntese à esquerda — LEFT$("(",1). A linha 10 concatenará esse parêntese (se o número for negativo) ou um espaço em branco (se o número for positivo) a cadeia de saída. A expressão **LEFT$ (" ", ABS (V # > 0))** se encarrega da operação, que depois é repetida para o parêntese direito.

DINHEIRO À BRASILEIRA

Examinaremos agora uma função capaz de formatar valores usando pontos e vírgulas, segundo a convenção bancária brasileira. Símbolos especiais representarão números positivos (créditos) e números negativos (débitos).

Como essa função é mais complexa, definiremos uma sub-rotina:

```
110 INPUT "Numero de colunas ";
E%
120 INPUT "Caractere de preench
imento ";B$
130 INPUT "Simbolo p/valor posi
tivo ";P$
140 INPUT "Simbolo p/valor nega
tivo ";N$
150 INPUT "Valor ";V#
160 GOSUB 1000·PRINT SS·COTO 15
0
1000 IS=MIDS(STRS(FIX(V#)),2)·F
S=MIDS(STRS(INT(.00*(V# FIX(V#)
))),2) SS=""
1010 IF LEN(IS)<=3 AND VAL(FS)>
0 THEN SS IS+SS+","+FS GOTO 103
0
1015 IF LEN(IS)<=3 AND VAL(FS)=
0 THEN SS=IS+SS GOTO 1030
1020 SS="."+RICHTS(IS,3)+SS IS=
LEFTS(IS,LEN(IS)-3)·GOTO 1010
1030 IF V#<0 THEN XS=NS ELSE XS
=PS
1040 SS=RIGHTS(STRINGS(E%,BS)+S
S+XS,E%):RETURN
```

Os argumentos de entrada são: **V #**, valor monetário em precisão dupla (incluindo centavos ou não, após o ponto); **E%**, número de colunas para formatação; **B$**, caractere de preenchimento à esquerda (asteriscos, zeros ou um espaço em branco); **P$** e **N$**, símbolos à direita do valor, para indicar créditos e débitos, respectivamente. O string de saída retorna em **S$**. As linhas 110 a 160 testam a sub-rotina.

Na linha 1000, **V #** é quebrado em dois valores inteiros, correspondentes aos cruzados e aos centavos. Esses valores são armazenados em duas cadeias, **I$** e **F$**, respectivamente.

As linhas 1010 e 1015 verificam se **I$** tem três ou menos dígitos (um valor igual ou menor que 999 cruzados), e se existe um valor em centavos **(VAL(F$) > 0)**. Conforme o resultado dos dois testes, a rotina termina, saltando para a linha 1030, que concatena o símbolo adequado para créditos e débitos ao final de **S$**. A linha 1040 enquadra o string **S$** de saída dentro de **E%** espaços. O laço formado pela linha 1020 reduz progressivamente o tamanho de **I$**, separando cada grupo de três dígitos por um ponto, até que **LEN(I$)** seja menor ou igual a 3.

Entre as aplicações dessa rotina, inclui-se o preenchimento de cheques os caracteres de proteção de campo são colocados à esquerda do valor. Se quiser adicionar um cifrão (ou o símbolo Cz$) entre os caracteres de proteção e o valor numérico, na linha 1000 da rotina, que inicializa S$, substitua a expressão **S$ = ""** por **S$ = "Cz$"**.

IMPRIMA SEUS DESENHOS

- TIPOS DE IMPRESSORA
- DESPEJO DE TELA
- PROGRAMAÇÃO GRÁFICA
- MONTAGEM DA TELA
- PROBLEMAS COM CORES

Ao desligar o micro, os maravilhosos desenhos que você criou na tela desaparecem sem deixar rastros. Com umas poucas linhas em BASIC, você poderá preservá-los em papel.

A capacidade gráfica dos micros permite a elaboração de vários tipos de gráfico ou desenho no vídeo. Quer você esteja interessado nas aplicações práticas desses gráficos, quer seja um artista preocupado em adotar um novo meio de expressão, as imagens produzidas com a ajuda do computador apresentam uma grande limitação: existem apenas enquanto a tela não for apagada. Se você quiser uma cópia permanente da imagem criada, precisará usar uma máquina fotográfica ou copiar a tela em uma impressora gráfica.

Para guardar uma listagem, uma lista de números ou um texto produzido por um programa, os usuários também têm que recorrer a uma impressora. Há, porém, uma grande diferença entre copiar uma tela de textos, escritos em ASCII padrão, e uma tela gráfica, de baixa ou alta resolução. Com exceção dos microcomputadores da linha Sinclair (ZX-81 e Spectrum), que possuem o comando **COPY**, os comandos de impressão do BASIC, como **LPRINT**, **PR#1**, **PRINT#-2** etc., funcionam apenas com caracteres ASCII. Alguns fabricantes oferecem impressoras que têm o conjunto ampliado de caracteres. Este inclui os caracteres semigráficos, típicos dos micros TRS 80, TRS-Color, TK-2000 e MSX, mas ainda assim não é suficiente para copiar uma tela com gráficos de média ou de alta resolução.

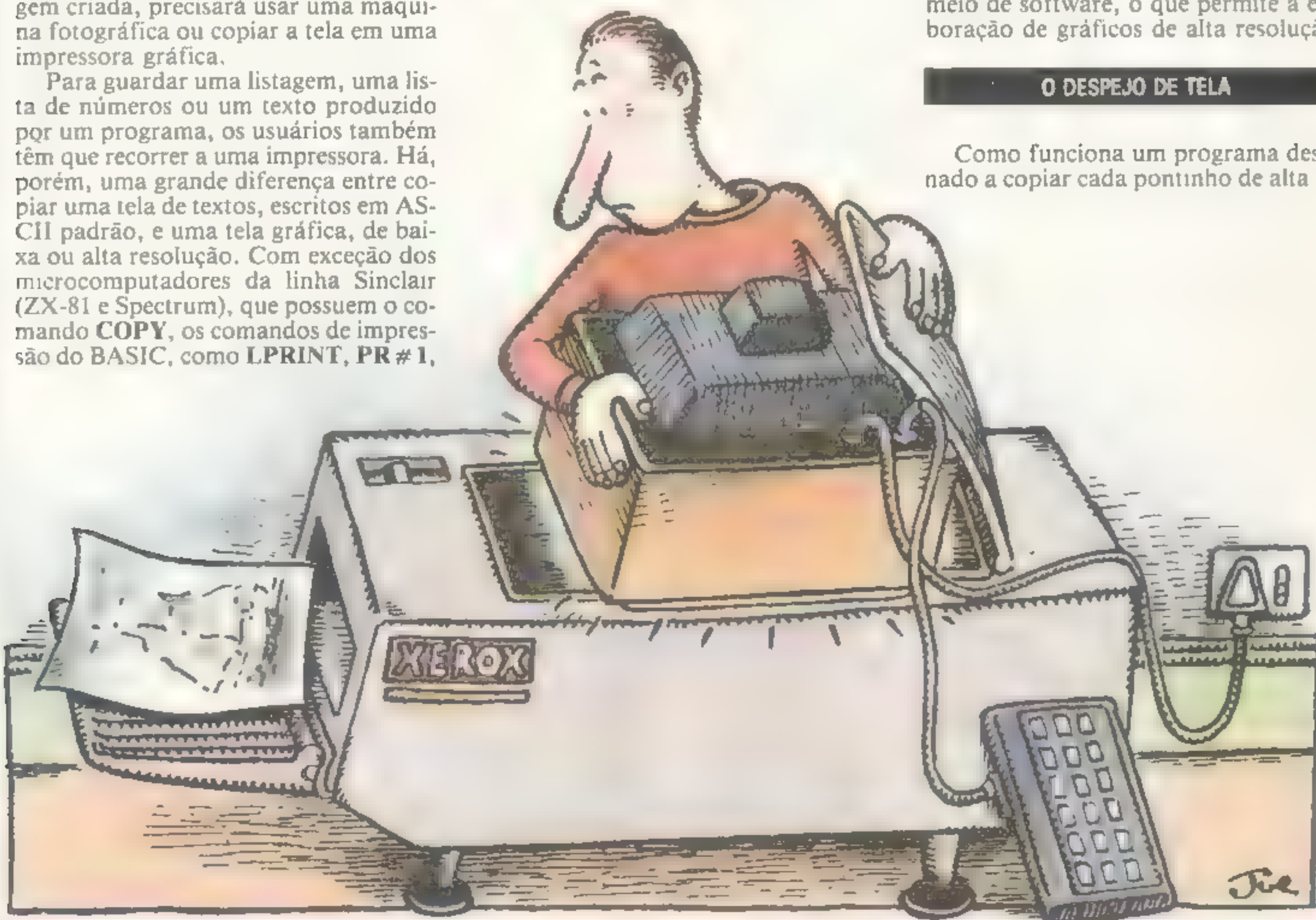
Como vimos em um artigo da seção *Periféricos* (página 648), existem dois modelos de impressora para micros: as impressoras de tipo formado (como as baseadas na "margarida"), e as impressoras matriciais. Embora algumas impressoras margarida possam fazer gráficos compostos de pontos, elas são muito lentas para ter alguma utilidade na cópia frequente de telas gráficas.

As impressoras matriciais, por sua vez, imprimem os caracteres utilizando um conjunto de pontos, num processo idêntico ao da formação de caracteres na tela de vídeo. O sinal enviado pelo computador à impressora aciona uma série de pinos ou agulhas situados na cabeça de impressão, de modo a reproduzir um padrão de pontos que compõe um determinado caractere.

Nas impressoras matriciais não-gráficas (impressoras de texto), os padrões formados para cada caractere já estão pré-programados em uma memória ROM na impressora, e não podem ser manipulados ou alterados individualmente. Nas impressoras matriciais gráficas, porém, pode-se programar cada agulha da cabeça de impressão por meio de software, o que permite a elaboração de gráficos de alta resolução.

O DESPEJO DE TELA

Como funciona um programa destinado a copiar cada pontinho de alta re-



solução de uma tela gráfica? O comando **COPY** dos micros da linha Sinclair e um bom exemplo disso. Esse comando efetua uma operação que chamamos *despejo de tela* (*screen dump*, em inglês).

Com o micro acoplado a uma impressora adequada (própria para a linha Sinclair), o comando **COPY** "traduz" as configurações de pontinhos de cada setor da tela em padrões de atuação das agulhas da cabeça de impressão. Desse modo, o conteúdo da tela (texto e/ou gráfico) e fielmente reproduzido — menos a cor, evidentemente.

Os sistemas operacionais de alguns micros (como o TRS-80) podem ter funções de despejo de tela, que também só atuam se uma impressora compatível estiver acoplada à máquina. O CP-500 da Prológica, por exemplo, despeja a tela semigráfica na impressora P-500S, quando as teclas J, K e L são pressionadas simultaneamente no teclado.

A maioria dos micros, porém, não tem comandos ou funções especiais para produzir despejos de tela. E mesmo que seu computador disponha desses recursos, lembre-se de que eles nada produzirão sem a impressora adequada.

PROGRAMAÇÃO DA IMPRESSORA

Não é possível escrever na impressora usando o mesmo método com o qual escrevemos na tela. Se dermos, por exemplo, o comando:

```
PRINT CHR$(255)
```

no TRS-Color, um bloco colorido será exibido na tela (255 é o código do caractere gráfico correspondente). Entretanto, se você tentar fazer isso com a impressora, enviando o comando:

```
PRINT#-2,CHR$(255)
```

nada acontecerá, pois a impressora interpretará o código 255 de modo totalmente diferente. Dizemos, por essa razão, que os *códigos de controle* dos dois periféricos, vídeo e impressora, produzem efeitos diferentes.

Precisamos, assim, de um programa especial para efetuar o despejo de tela — um programa que prepare a impressora para receber gráficos. Algumas funções automáticas da impressora devem ser desligadas, e outras, ligadas. É necessário acionar a cabeça de impressão de forma que as agulhas possam ser controladas individualmente. Além disso, o avanço horizontal e vertical da cabeça de impressão precisa ser feito em passos diminutos, sem deixar espaços entre cada ponto impresso.

As impressoras gráficas podem ser programadas externamente para efetuar as funções mencionadas. Nesse *modo gráfico*, é possível atuar sobre grupos de agulhas usando códigos de oito bits. A informação captada na memória de vídeo e enviada para a impressora não sob a forma de códigos ASCII, mas como códigos binários correspondentes ao padrão de ativação das agulhas da cabeça. Assim, existem códigos binários para desligar o avanço automático de linha, para avançar a cabeça de uma linha para outra, para definir o espaço entre as linhas etc. O programa propriamente dito simplesmente "varre" a tela e produz as linhas de impressão correspondentes a cada grupo de bits.

Mais adiante, apresentaremos programas desse tipo para os micros Spectrum e TRS-Color. Antes disso, porém, precisamos examinar as combinações micro-impressora possíveis.

COMPATIBILIDADE

Os programas de despejo de tela dependem não só do tratamento dado à tela gráfica pelo computador, como também da marca e do modelo da impressora que será utilizada. Isso ocorre porque ainda não existe uma padronização dos caracteres de controle gráfico e da cabeça de impressão.

Nossos programas destinam-se às máquinas que adotam o padrão Epson. Este foi estabelecido para as impressoras do fabricante da marca Epson, e pode ser encontrado em modelos muito populares, como, por exemplo, o Epson MX-80, MX-120 e FX-80. No Brasil, várias empresas, entre elas a Elebra (impressoras Mônica e Alice), a Rima e a Grafix, seguem esse padrão.

As impressoras compatíveis com o padrão Epson usam uma matriz de oito pontos de altura. Esta é a configuração mais conveniente, pois permite ao micro enviar informações provenientes da tela, um byte de cada vez, pela interface paralela. O TRS-Color possui interface serial, mas o método é o mesmo. No Spectrum e no ZX-81, o usuário deve inserir uma interface paralela no conector de expansão. O Spectrum, além disso, precisa ser carregado com um programa em linguagem de máquina, para usar a impressora.

TEORIA DE PROGRAMAÇÃO GRÁFICA

A programação da sequência de impressão das agulhas da cabeça para a obtenção de efeitos gráficos pode ser feita por meio de alguns comandos simples em BASIC. O processo é muito fácil. Para entendê-lo, devemos examinar como certos bytes enviados para a impressora são interpretados pelos circuitos de controle da mesma.

A impressora normalmente está capacitada para reconhecer os códigos ASCII e ASCII estendido, que estão pré-programados na memória ROM. Assim, se enviarmos uma sequência de códigos entre hexadecimal 20 (correspondente a espaço em branco) e 7E (til), a impressora irá reconhecê-los como códigos de caracteres. Por exemplo, a palavra ABA é impressa quando enviamos:

```
LPRINT CHR$(65);CHR$(32),
CHR$(65)
```

Entre hexadecimal 00 e 1F situam-se caracteres de controle com diversas atribuições, que variam de impressora para impressora. No padrão Epson, **CHR$(7)** faz soar o alarme sonoro da impressora, **CHR$(13)** provoca um retorno de carro, enquanto **CHR$(12)** determina a mudança de página.

Outras funções de controle podem ser acionadas por meio da combinação de vários caracteres. Essas sequências começam sempre pelo caractere **CHR$(27)**, que corresponde ao **ESCAPE** — por isso, são chamadas de *sequências de escape*. A sequência **ESC G**, por exemplo, coloca a impressora em modo de impressão em qualidade carta (cada caractere é impresso duas vezes).

```
LPRINT CHR$(27);"G"
```

Para mudar o espaçamento entre linhas de impressão, utiliza-se **ESC A n**, onde **n** refere-se ao número de passos por avanço de linha:

```
LPRINT CHR$(27);"A";CHR$(4)
```

Um passo é equivalente a 1/48 ou 1/72 de polegada, dependendo do modelo da impressora (com densidade simples ou densidade dupla). No exemplo anterior, o deslocamento entre cada linha de impressão equivale ao espaçamento entre quatro agulhas.

A seleção do modo gráfico, no padrão Epson, é feita pela sequência **ESC K** ou **ESC L**. Quando você seleciona o modo gráfico, o código binário enviado à impressora aciona diretamente o padrão de agulhas da cabeça de impressão: quando o bit for 1, a agulha correspondente é acionada; quando o bit for 0, a agulha fica inativa. O bit menos significativo (D0) controla a agulha inferior, e o bit mais significativo (D7), a agulha superior. Assim, se quisermos imprimir uma coluna de pontos, a agulha de cima e as duas últimas agulhas de baixo não serão acionadas; o código a

enviar é 10000011, ou 128 + 0 + 0 + 0 + 0 + 0 + 2 + 1 = 131 em decimal.

A sequência de escape precisa especificar, ainda, o número de bytes de código gráfico que serão enviados, e, em seguida, os bytes propriamente ditos: **ESC K n1 n2 dados**.

Os bytes **n1** e **n2** são, respectivamente, o mais significativo e o menos significativo de um número de dezesseis bits, e devem ser apresentados no conhecido formato **256 * n2 + n1**.

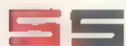
Se vamos enviar 550 bytes gráficos, damos o comando:

```
LPRINT CHR$(27);CHR$(38);CHR$
(2);
```

pois 256*2 + 38 = 550. A expressão que se segue, em BASIC, serve para calcular **n1** e **n2** a partir de **n**, que representa o número total de bytes:

```
N1 = INT (N/256)
N2 = N INT (N/256)
```

Este programa ilustra o padrão de acionamento das agulhas:



```
10 LPRINT CHR$(27);"A";CHR$(8)
20 LPRINT CHR$(27);"K";CHR$(16)
;CHR$(0);
30 FOR J=7 TO 0 STEP -1
40 LPRINT CHR$(2**J);
50 NEXT J
60 FOR J=0 TO 7
70 LPRINT CHR$(2**J);
80 NEXT J
90 LPRINT CHR$(10);
```



```
10 LPRINT CHR$(27);"A";CHR$(8)
20 FOR I=1 TO 5
25 LPRINT CHR$(27);"K";CHR$(16)
;CHR$(0);
30 FOR J=7 TO 0 STEP -1
40 LPRINT CHR$(2^J);
50 NEXT J
60 FOR J=7 TO 0 STEP -1
70 LPRINT CHR$(2^J);
80 NEXT J
90 LPRINT CHR$(10);
100 NEXT I
```



```
10 PRINT#-2,CHR$(27);"A";CHR$
(8)
20 PRINT#-2,CHR$(27);"K";CHR$
(16);CHR$(0);
30 FOR J=7 TO 0 STEP -1
40 PRINT#-2,CHR$(2^J);
50 NEXT J
60 FOR J=7 TO 0 STEP -1
70 PRINT#-2,CHR$(2^J);
80 NEXT J
90 PRINT#-2,CHR$(10);
```



```
5 PR#1
10 PRINT CHR$(27);"A";CHR$(8)
20 PRINT CHR$(27);"K";CHR$(16);
CHR$(0);
30 FOR J=7 TO 0 STEP -1
40 PRINT CHR$(2^J);
50 NEXT J
60 FOR J=7 TO 0 STEP -1
70 PRINT CHR$(2^J);
80 NEXT J
90 PRINT CHR$(10);
100 PR#0
```

A linha 10 estabelece um espaçamento de oito passos entre linhas. A 20 programa a impressora para o modo gráfico, avisando que a seguir serão enviados dezesseis bytes gráficos. Quando os dados não estão incluídos imediatamente na sequência de escape, o ponto e vírgula no final dessa linha é obrigatório. Os laços das linhas 30 a 50 e 60 a 80 imprimem, respectivamente, um padrão descendente e outro ascendente de pontos (determinados pelos códigos que correspondem a potências de 2: 128, 64, 32 etc.). Finalmente, a linha 90 alimenta uma linha.

MONTAGEM DA TELA

Para copiar um desenho na impressora, é necessário executar um programa que o trace na tela. Existem duas alternativas para isso. Uma delas consiste em carregar o programa de desenho no computador e, em seguida, combiná-lo com o programa de despejo (a numeração das linhas de ambos tem que ser diferente). O primeiro programa deve ser alterado na tela depois que o desenho estiver pronto, de modo que o programa de despejo seja chamado automaticamente, ou quando se pressionar determinada tecla. Se achar conveniente, poderá incluir permanentemente o programa de despejo no programa de desenho, na forma de uma sub-rotina.

A segunda alternativa é a de mais fácil execução, mas depende da capacidade do micro de armazenar o conteúdo de telas gráficas em disco ou fita (comandos **BSAVE**, **SAVEM** etc.). Se seu computador tem essa capacidade, carregue e execute o programa de desenho e armazene a tela resultante em fita ou disco. Depois, carregue o programa de despejo usando um comando na linha 10 (essa linha não foi colocada nos programas que se seguem justamente para criar um espaço para o seu comando de carregamento). Com esse procedimento, o programa irá ler, em primeiro lugar, a tela armazenada.

Para os usuários do TRS-Color, o sistema é um pouco diferente: algumas linhas do programa de despejo têm que ser executadas antes que o desenho seja traçado na tela.

IMAGEM DE LADO

A imagem despejada por este programa é imprimida de lado — ou seja, no sentido longitudinal e não transversal, relativamente à imagem que aparece na tela. Com isso, obtemos uma imagem maior e mais margem no papel.



```
15 LPRINT CHR$5;
20 LPRINT CHR$27; "A"; CHR$8;
30 FOR x=0 TO 255 STEP 4
40 LPRINT CHR$13, CHR$27; "K",
CHR$0, CHR$96, CHR$1,
50 FOR y=0 TO 175 STEP 8
60 FOR d=0 TO -7 STEP -1
70 LET bt=(POINT(x,y-d)*128)+(P
OINT(x,y-d)*64)+(POINT(x+1,y-d)
*32)+(POINT(x+1,y-d)*16)+(POINT
(x+2,y-d)*8)+(POINT(x+2,y-d)*4)
+(POINT(x+3,y-d)*2)+(POINT(x+3,
y-d))
72 LPRINT CHR$bt,
74 LPRINT CHR$bt,
80 NEXT d
90 NEXT y
100 NEXT x
110 LPRINT CHR$4: STOP
```

A linha 15 desliga o conjunto de caracteres ASCII na impressora. O comando **LPRINT** da linha 20 informa à impressora que não deve haver espaçamento entre as linhas impressas.

O laço que vai da linha 30 à 100 varre uma linha da tela, em blocos de quatro pixels de cada vez. A linha 40 passa para a impressora um código de retorno de carro (código 13) e, em seguida, o código *0, que se encarrega de preparar a máquina para receber o número de bytes que a linha corrente de tela irá enviar. Esse byte é composto dos códigos 96 e 1 (também na linha 40), que são interpretados como 1*256 + 96 — ou seja, 352 bytes.

O laço da linha 50 à linha 90 percorre a tela de cima para baixo; o laço da linha 60 à 80 toma um bloco de oito pixels de cada vez. Antes de incrementar os laços, a linha 70 usa o comando **POINT** para ler os pixels na tela. Estes são condensados pela expressão numérica no byte **bt**, que é enviado duas vezes à impressora, pelas linhas 72 e 74. A segunda impressão fará a imagem aparecer duas vezes maior que na tela, na direção horizontal.

Finalmente, a linha 110 retorna a impressora ao conjunto ASCII e encerra

a execução. Se você quiser transformar esse programa em sub-rotina, não se esqueça de substituir o **STOP** dessa linha por um **RETURN**.



```
2 DIM A(8,1)
4 FOR K 0 TO '  REA  A(K,0).A(K
.1) NEXT
6 DATA 3,3.2.1.0.0 3.1.3.3.0.0.
1,2.3.1.3.3
20 PRINT # 2. CHRS(27) 'A".CHRS
(8)
30 FOR L=0 TO 255 STEP 4
40 PRINT #-2,CHRS(13);CHR$(27).
"K".CHRS(0) CHRS(128 ,CHRS 1),
50 FOR K=191 TO 0 ST P  1
60 T=0 S 0 FOR M 0 TO 3 P PPOIN
T(L+M,K) T=T*4+A(P,0) S S*4+A(P
.1):NEXT
70 PRINT # 2,CHRS(T),CHRS(S
80 NEXT K.L
90 PRINT # 2,CHRS(27),"@"
```

É importante que as três primeiras linhas do programa (2. 4 e 6) precedam a rotina de geração da tela gráfica que será impressa. Na linha 10, reservada para o seu programa, você deve estabelecer o tipo de **SCREEN** e de **PMODE**. Em virtude da forma de armazenagem de telas gráficas no TRS-Color, o programa de despejo precisa levar em conta as cores empregadas.

A linha 2 dimensiona um conjunto **A**, para armazenar o padrão de pontos usado em cada cor. Existem nove cores, mas nem todas podem aparecer na tela ao mesmo tempo. Os padrões estão definidos em **DATA**, na linha 6, e são carregados no conjunto **A** pela linha 4. Dois itens em **DATA** determinam cada cor; o segundo é impresso sobre o primeiro. Por exemplo, o primeiro par de números (3,3) indica preto, formando um padrão binario de 3 (11) sobre 3 (11). O segundo par (2,1) especifica verde, formando um padrão de 1 (01) sobre 2 (10). Note que há repetição de pares na sequência, mas isso não importa: a segunda ocorrência determina uma cor que não pode aparecer na tela junto com a primeira especificação.

A linha 20 prepara a impressora para receber gráficos. A sequência de escape que contem instrui a impressora a não dar espaçamento entre linhas.

O laço que vai da linha 30 à 80 percorre uma linha da tela, em blocos de quatro pixels. A linha 40 envia à impressora um codigo de retorno de carro (13) e, em seguida, o código *, que coloca a impressora em modo gráfico. Ainda nessa linha, **CHR$(0)** define a densidade de impressão e **CHR$(128);CHR$(1)** diz à impressora para esperar 128 + 1*256 bytes gráficos, ou seja, um total de 384 bytes.

O laço da linha 50 à 80 percorre a tela de cima para baixo (192 pixels de altura); o da linha 60 toma quatro pixels de cada vez, examina-os com **PPOINT**, calcula as cores e coloca o resultado nos bytes **S** e **P**. A linha 70 imprime esses bytes. Finalmente, a linha 990 reinicializa a impressora.

PROBLEMAS COM CORES

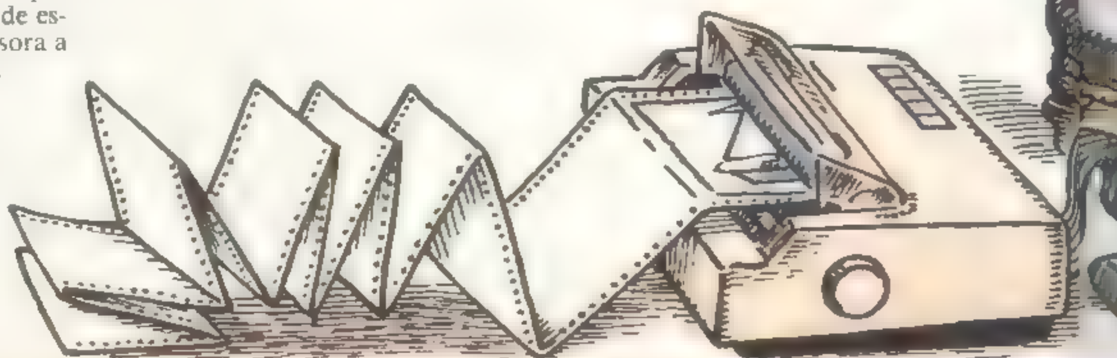
O programa de despejo de tela, em BASIC, imprime apenas uma representação binária, em preto e branco, da tela de alta resolução, pois reproduz o padrão de pixels sem diferenciar as cores presentes na imagem original. Em conseqüência, a imagem obtida apresenta densidade igual para todas as cores, não representando com fidelidade o desenho copiado.

É possível, porém, escrever um programa de despejo de tela em que as cores sejam representadas por diferentes tonalidades de cinza e preto. Esse efeito é conseguido por impressões sobrepostas de partes da imagem, de modo a obter tons mais escuros ou claros de cinza, conforme a cor presente. Em uma tela com quatro cores, por exemplo, podem-se definir quatro diferentes padrões de pontos, para representar o vermelho, com um cinza bem claro; o verde, com um cinza mais escuro etc. Isso é feito inibindo-se o avanço da cabeça de impressão.

Evidentemente, o tempo total de impressão é muito longo, nesse caso. Um despejo de tela simples demora quase meia hora em alguns micros; se levarmos em consideração a cor, esse tempo será aumentado para várias horas! Um programa em código de máquina — como o que apresentamos a seguir para duas linhas de computadores — pode resolver esse problema.



```
10 CLEAR 59999
20 LET L=100  RESTORE L· FOR N=
60000 TO 60247 STEP 8
30 LET T=0:FOR M=0 TO 7
40 READ A·LET T=T+A·POKE N+M.A·
NEXT M
50 READ A IF A<>T THEN PRINT "E
RRO NOS DADOS DA LINHA ".L. STO
P
60 LET L=L+10: NEXT N : STOP
100 DATA 243.62.3.205,1.22.33.7
0.639
110 DATA 235,6.4.205.9.235,62.0
.756
120 DATA 50.83.235.62.0.50.84.2
35.799
130 DATA 6.175.221.33.85,235.19
7.62.1014
140 DATA4,237.75.83.235.245.205
.15.1099
150 DATA 235,245.205.30,235.241
.126,32.1349
160 DATA 6.203.63,203.63.203.63
.230,1034
170 DATA 7.214.7.237.68.221.114
.0.873
180 DATA 221.35.241.12,61.32,22
2,58.882
190 DATA 84,235.660.50.84,235.1
93.16,957
200 DATA 205,6,7,197.33,74.235.
6.763
210 DATA 9.205 4.235.221 33.85
235.1032
220 DATA 6.175.197,6.1.30.0.197
.615
230 DATA203.35.203.35,221.1.2 3
.254,1077
240 DATA 0.40.7.221.53.0.62.3.3
86
250 DATA 24.2.62,0,131.95.221.3
5,570
260 DATA 193.166.228.123.245.21
5,241.215.1476
```



```
270 DATA 193,16,215,193,16,197,
33,65,928
280 DATA 235,6,5,205,9,235,62,1
0,767
290 DATA 215,58,83,235,198,4,50
,83,926
300 DATA 235,210,115,234,62,4,2
15,251,1326
310 DATA 201,126,35,215,16,251,
201,197,1242
320 DATA 205,170,34,71,4,126,20
3,7,820
330 DATA 16,252,130,1,143,201,1
97,62,1152
340 DATA 175,144,230,248,71,88,
21,0,978
350 DATA 203,35,203,18,203,35,2
03,18,923
360 DATA 121,203,63,203,63,203,
63,111,1030
370 DATA 38,0,25,17,0,88,25,193
,386
380 DATA 201,27,64,27,65,8,5,27
,424
390 DATA 65,8,27,65,0,13,27,42,
247
400 DATA 0,96,1,0,0,48,48,193,3
86
```



```
10 CLEAR 200,29992
20 CLS:FOR K=0 TO 12:T=0:FOR L=
0 TO 23:READ A
30 POKE 29993+24*K+L,A:T=T+A
40 NEXT:READ A:IF A<>T THEN PRI
NT "ERRO NOS DADOS DA LINHA",10
00+10*K:END
50 NEXT
1000 DATA 0,0,0,3,27,51,24,134,
254,151,111,111,140,242,111,140
,240,150,182,133,1,39,3,108,235
5
1010 DATA 140,231,77,39,4,129,2
,38,3,108,140,220,150,193,68,16
7,140,216,48,140,214,23,0,142,2
632
1020 DATA 95,52,4,51,141,1,1,19
8,191,166,228,52,6,134,4,52,2,2
3,0,134,48,140,107,166,1996
1030 DATA 134,167,192,108,97,10
6,228,38,240,53,2,53,6,166,228,
90,193,255,38,223,198,3,52,6,28
76
1040 DATA 51,141,0,212,198,192,
231,228,48,140,65,141,81,79,198
,4,72,72,106,192,43,2,138,3,263
7
1050 DATA 90,38,245,173,159,160
,2,173,159,160,2,106,228,38,230
,134,13,173,159,160,2,106,97,38
,2845
1060 DATA 207,53,6,134,10,173,1
59,160,2,53,4,173,159,160,0,129
,3,39,4,203,4,38,138,48,2059
1070 DATA 140,17,32,18,5,27,42,
4,128,1,3,1,0,2,3,0,1,2,3,2,27,
64,230,128,880
1080 DATA 166,128,173,159,160,2
,90,38,247,57,134,32,109,141,25
5,48,39,1,68,52,2,166,101,214,2
582
1090 DATA 182,193,1,34,1,68,230
,224,61,211,186,31,1,230,99,84,
84,84,109,141,255,18,39,1,2567
1100 DATA 84,58,166,99,109,141,
255,8,39,8,132,15,64,139,15,68,
32,5,132,7,64,139,7,198,1984
1110 DATA 1,74,43,3,88,32,250,1
09,141,254,238,39,14,52,4,197,8
5,39,5,88,235,224,32,3,2250
1120 DATA 84,235,224,52,4,166,1
32,164,224,84,37,3,68,32,250,17
1,141,254,206,171,141,254,203,5
7,3357
```

Embora o código de máquina esteja contido em uma lista de números em **DATA**, cada linha inclui somas de verificação, cuja finalidade é evitar que se cometam erros de cópia.

Não se esqueça de observar as precauções usuais ao rodar um programa em linguagem de máquina: para não correr riscos, armazene o programa, assim como suas versões corrigidas, em fita ou disco, antes de executá-lo.

Depois de testar o programa completo, rode-o com **RUN** e, em seguida, apague-o (o programa em código de máquina ficará armazenado em uma porção protegida da memória RAM). Digite ou carregue seu programa de desenho e execute-o. Para imprimir a imagem na tela, chame este comando:



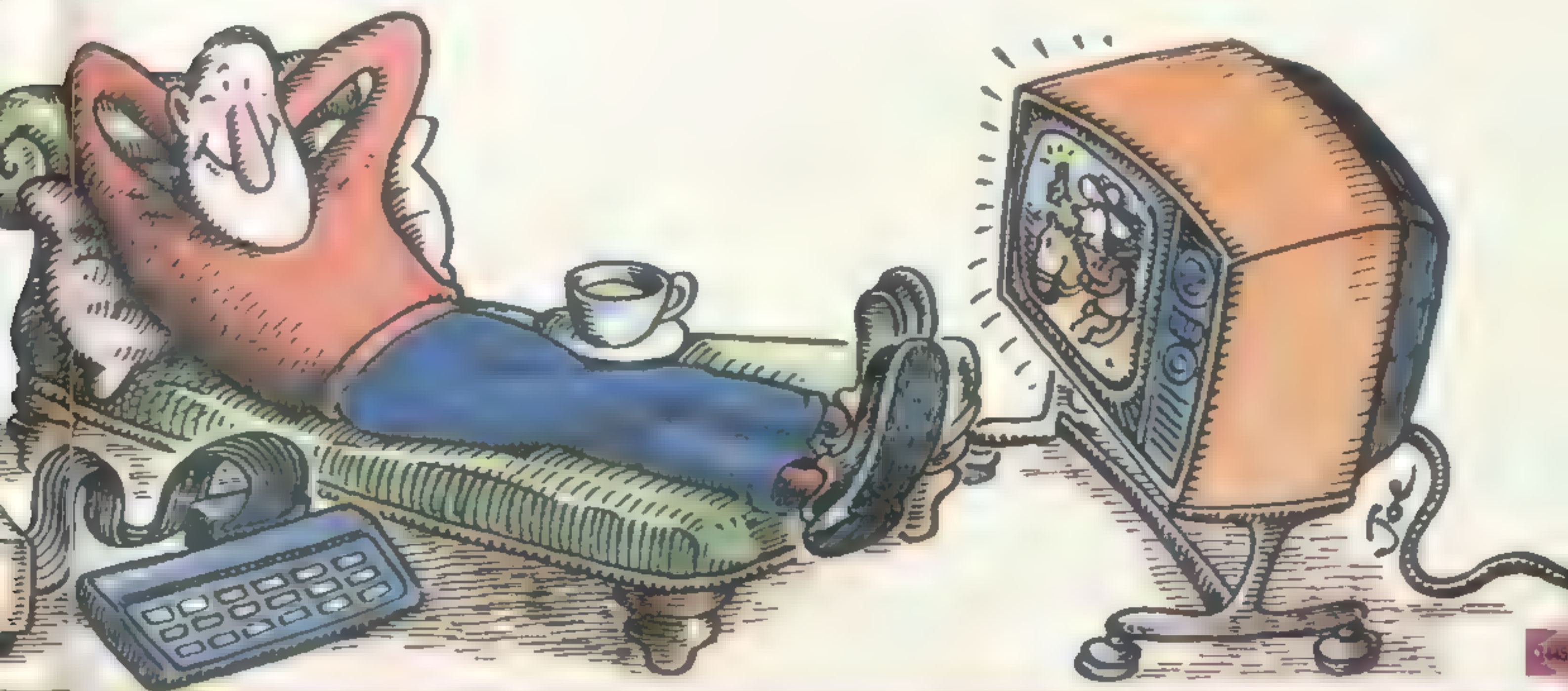
```
RANDOMIZE USR 60000
```



```
EXEC 30000
```

É importante desligar a alimentação automática de linha em sua impressora, antes de usá-la para despejos de tela. Para isso, siga as instruções do manual de operação.

Imprima a tela em uma fita de impressão já usada, pois isso proporcionará um contraste melhor entre as diferentes tonalidades.



ESTRUTURAS DO PASCAL

Já examinamos a filosofia e os princípios básicos do Pascal. Mostraremos agora como as estruturas dessa linguagem são organizadas na construção de um programa.

[illegible] te a um programa que possa ser com- [illegible]

Para realizar essa tarefa, você preci- [illegible] pelo Pascal. Esse procedimento não é muito diferente daquele [illegible] BASIC [illegible] crocomputador.

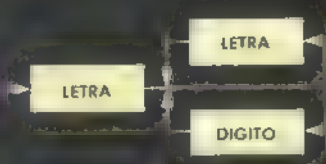
No BASIC [illegible] FOR...NEXT [illegible] IF...THEN [illegible] BASIC [illegible]

COMPATIBILIDADE

[illegible] BASIC [illegible] MSX ou do TK-2000, por exemplo [illegible] programas [illegible] compilador [illegible]

Pela mesma razão [illegible] programa [illegible] determinada forma para cada aplicação. Para apresentar o padrão [illegible] definição da forma e da [illegible] do programa, pode-se recorrer à [illegible] desenvolvida durante os anos 60, conhecida como Forma de Backus-Naur (BNF), ou ao diagrama de sintaxe [illegible]

[illegible] ção. Suponhamos que se queira definir um identificador: um identificador definido como uma letra ou um dígito [illegible]



[illegible] ... da definição dada anteriormente. Para definir [illegible] ... BNF [illegible]

[illegible]

[illegible] ... BNF, para mostrar sua [illegible] ... BASIC [illegible]

TECNICAS DE REPETICAO

[illegible] três técnicas de repetição [illegible] Pascal. A escolha de uma delas depende do parâmetro a ser empregado no controle do laço. A primeira é familiar aos usuários do BASIC: **FOR...DO**. As outras duas, que também podem ser simuladas naquela linguagem, são: **WHILE...DO** e **REPEAT...UNTIL**.

A principal diferença [illegible] está na condição de controle [illegible] ponsável pelo número de repetições. Na forma **WHILE...DO** (ENQUANTO...FAÇA), as instruções seguintes [illegible] executadas enquanto determinada [illegible] pressão continuar sendo verdadeira. [illegible] forma **REPEAT...UNTIL** (REPITA...ATÉ QUE) irá executar uma série de instruções até que uma dada condição [illegible] já alcançada. Ambas as técnicas são [illegible] das quando não se sabe de antemão o número de repetições necessárias. Caso [illegible] número possa ser previamente defini[illegible], utiliza-se a forma **FOR...DO**.

WHILE...DO

Em BNF, [illegible]

[illegible]

[illegible] laço é empregado quando o número de repetições depende de uma condição. Assim, será executado enquanto a condição especificada no programa for verdadeira. Para repetir uma série de instruções dentro desse laço, adiciona-se, após o do, um begin — indicando o início das instruções — [illegible] end — indicando o término do laço.

O exemplo que se segue mostra [illegible] do **WHILE...DO**. Tente escrever o [illegible] [illegible]

```
program exemplo7;
var no,sum:integer;
begin
  sum:[illegible]
  read(no);
  while no[illegible]
  begin
    sum:=[illegible]
    read(no)
  [illegible]
  [illegible]
end
```

REPEAT...UNTIL

repeat < instrução > **until** < expressão lógica >

As instruções entre **repeat** e **until** são repetidas até que a condição final se torne verdadeira. Todas elas são executadas pelo menos uma vez, pois o teste só é feito ao final. Ao contrário da forma **while...do**, não é necessário acrescentar **begin** e **end** a um conjunto de instruções. Usando-se essa técnica, o programa anterior pode ser escrito da seguinte maneira:

```
program exemplo3;
var no,sum integer;
begin
    sum:=0;
    repeat
    read(no);
    sum:=sum+no;
until no=0;
write(sum)
end
```

Esse programa é mais eficiente que o anterior, pois, com o uso de **repeat...until**, economiza-se uma instrução **read(no)**. Para efetuar uma soma corrente, é melhor que o programa entre diretamente no laço inicializado por **repeat**. No programa **exemplo2**, o programa chega à solução por intermédio de uma estrutura menos adequada que a utilizada no **exemplo3**.

FOR

No Pascal, ao contrário do que acontece no BASIC, o laço inicializado por **FOR** só pode caminhar em passos (**STEP**) de + 1 ou − 1. Adaptando o programa anterior ao uso dessa técnica, teremos que introduzir inicialmente a quantidade de números a serem somados.

```
program exemplo4;
var no,sum,quant,i:integer;
begin
    read(quant);
    sum:=0;
    for i:=1 to quant do
    begin
        read(no);
        sum:=sum+no
    end;
    writeln(sum)
end.
```

TOMANDO DECISÕES

Como o BASIC, o Pascal dispõe de vários recursos para verificar qual decisão deve ser tomada diante de determinado resultado. O primeiro deles, comum ao BASIC, é o **if...then...else**, geralmente usado quando a seleção depende de um ou outro de dois eventos. Porém, se a decisão depende de um número maior de eventos, costuma-se utilizar **case...of**. Não há nenhuma estrutura similar a **case** no BASIC.

IF...THEN...ELSE

if < expressão lógica > **then** < instrução 1 > **else** < instrução 2 >

Essa estrutura indica que, se a expressão lógica for verdadeira, o programa executará a instrução 1; se for falsa, a instrução 2. Como mostra o diagrama FBN, o **else** é opcional; uma construção alternativa seria **if...then.**

CASE

A declaração **case** permite ao programa selecionar uma instrução de uma lista de várias possibilidades.

A forma geral de **case** é:

case expressão **of**

c1 : instrução;
c2 : instrução;
''
''
''
cn : instrução
end;

O valor da expressão deve corresponder a um dos **c** e ser do mesmo tipo. No Pascal padrão, se o valor da expressão não for encontrado na lista, uma mensagem de erro será exibida. Em algumas versões, porém, o programa simplesmente ignorará o **case**.

Note que há um **end** associado ao **case** que não corresponde a nenhum **begin**. Esse tipo de estrutura não é freqüente, e pode trazer inconvenientes durante a busca de um erro dentro do programa. Geralmente, a primeira coisa que se faz é verificar se há correspondência entre o número de **begin** e **end**. A associação do **end** ao **case** provocaria uma diferença. Para contornar o problema, coloca-se um rótulo em cada **end** — por exemplo, **end;** {**case**}.

As instruções equivalentes a cada opção (**c**) podem ser instruções compostas, sempre entre um **begin** e um **end.**

Veja este exemplo do uso de **case**:

```
program exemplo5;
var nodia:integer;
begin
 readln(nodia);
 if (nodia>0) and (nodia<8)
 then
   case nodia of
   1:writeln('Segunda-feira');
   2:writeln('Terça-feira');
   3:writeln('Quarta-feira');
   4:writeln('Quinta-feira');
   5:begin
    writeln('Sexta-feira');
    writeln('Dia de pagamento')
     end;
   6,7:begin
       writeln('Fim de-semana');
       writeln('Descanso');
       end;
   end
  else begin
       writeln('Use valores');
       writeln('entre 1 e 7')
       end;
end.
```

Em algumas variações do Pascal, os rótulos 1, 2, 3, 4, 5, 6 e 7 devem ser colocados entre parênteses, fugindo, portanto, do Pascal padrão. Na verdade, em algumas implementações para a adaptação ao sistema BASIC, a mudança na sintaxe é necessária.

O programa **exemplo5** mostra como superar um problema bastante freqüente no uso do **case**. Muitas vezes, o identificador pode assumir valores não encontrados entre os rótulos de **case**, o que provocaria uma interrupção e uma mensagem de erro. Assim, utilizamos um desvio condicional **if...then...else** para mudar o curso do programa, caso os valores de **nodia** não se encontrem no intervalo apropriado. Observe também que foram empregados rótulos com mais de um valor e instruções compostas após alguns deles.

DESENVOLVIMENTO DE PROGRAMAS

Com um conhecimento mais detalhado dos procedimentos disponíveis no Pascal, podemos passar ao exame de um programa escrito nessa linguagem.

O programa que apresentamos a seguir verifica se uma palavra ou frase é um palíndromo — ou seja, se ela pode ser lida de trás para frente sem sofrer alteração (como "radar" ou "Socorram-me, subi no ônibus em Marrocos"), ignorando-se os espaços em branco e os sinais de pontuação. Suponhamos que você queira testar se esta frase é um palíndromo:

Arara é ave rara

O algoritmo inicial poderia ser:

begin
leia uma cadeia de caracteres
teste se é um palíndromo
end

Essa frase inicial é bem simples, consistindo só de duas etapas, além do **begin** e do **end** requeridos pelo Pascal. Poderíamos elaborar um algoritmo mais detalhado para a primeira etapa:

início
leia o número de caracteres (n)
leia os caracteres colocando-os na matriz A(i)
fim

Traduzindo essa primeira etapa para o Pascal, temos:

```
readln(n)
for i:=1 to n do read(a[i]);
```

Passando para a segunda etapa, chegamos ao seguinte algoritmo:

início
faça crsc igual a 1
faça decr igual a n
enquanto crsc < decr faça
se a[crsc] é pontuação então
incremente crsc
senão
se a[decr] é pontuação então
decremente decr
senão
se a[crsc] = a[decr] então
incremente crsc
decremente decr
senão
faça a variável booleana (pal) igual a *false*
fim

Esse algoritmo poderia ser refinado para o Pascal desta maneira:

```
crsc:=1;
decr:=n;
while (crsc < decr) and pal do
if(a[crsc]=' ')or(a[crsc]='.')
or(a[crsc]=',')or(a[crsc]='"')
 then crsc:=crsc+1
 else
 if(a[decr]=' ')or(a[decr]='.')
 or(a[decr]=',')or(a[decr]='"')
  then decr:=decr-1
  else
   if a[crsc]=a[decr] then
   begin
   crsc:=crsc+1;
   decr:=decr-1;
   end
   else pal:=false
```

Precisaríamos de uma terceira etapa, na qual o programa mostraria o seguinte resultado:

se pal é igual a *true* então
escreva "palíndromo"
senão
escreva "não é palíndromo"

o que, em Pascal, corresponderia a:

```
if pal then
writeln('palindrome');
else
writeln('nao há palíndrome')
```

O PROGRAMA COMPLETO

Refinando algumas partes anteriores e reunindo-as, chegamos ao seguinte programa em Pascal:

```
program exemplo6;
const comp=30;
var a:array [1..comp] of char;
    i,n,crsc,decr : integer;
    pal : boolean;
begin
pal:=true;
readln(n);
for i:=1 to n do read(a[i]);
crsc:=1;
decr:=n;
while (crsc < decr) and pal do
if(a[crsc]=' ')or(a[crsc]='.')
or(a[crsc]=',')or(a[crsc]='"')
 then crsc:=crsc+1
 else
 if(a[decr]=' ')or(a[decr]='.')
 or(a[decr]=',')or(a[decr]='"')
  then decr:=decr-1
  else
   if a[crsc]=a[decr] then
   begin
   crsc:=crsc+1;
   decr:=decr-1;
   end
   else pal:=false;
if pal then
writeln('palindrome')
else
writeln('nao ha palindrome')
end.
```

O Pascal apresenta um tipo de variável desconhecido do BASIC, a variável *booleana*, que não assume valores numéricos nem guarda caracteres. Ela assume o valor de resultados lógicos — *true* (verdadeiro) e *false* (falso). Note que é desnecessário dizer:

if pal = true then...

bastando apenas:

if pal then...

REFINANDO O PROGRAMA

Se o seu sistema Pascal dispõe de algumas formas adicionais de manipular dados, diversos aperfeiçoamentos podem ser incorporados ao programa. Entretanto, como está, ele deve funcionar na maioria dos sistemas.

A variável booleana é usada para desviar o programa do laço **while** quando se determina que **a[crsc]** não é igual a **a[decr]**. Tudo o que precisa ser feito é tornar **pal** igual a *false* quando o teste **a[crsc] = a[decr]** apresentar esse resultado. No momento em que o programa voltar para o cabeçalho do laço, encontrará um desvio condicional dentro de **while**. A tecnica empregada em BASIC seria a seguinte:

```
140
150 FOR I=1 TO N


190 IF A(CRSC)=A(DECR) THEN
 260


230 NEXT I
240 PRINT "PALINDROME"
250 GOTO 270
260 PRINT "NAO E PALINDROME"
270 END
```

Um algoritmo melhor estruturado poderia ser apresentado assim:

início
defina o conjunto pontuação
...
...
se a[crsc] estiver em pontuação então
incremente crsc
senão
se a[decr] estiver em pontuação então
decremente decr
senão
...

Uma declaração desse tipo economizaria exaustivas comparações do tipo:

if (a[crsc] = ' ') or (a[crsc] = ',') or (a[crsc] = '.') or (a[crsc] = '"') then...

O Pascal aceita a definição de conjuntos (alguns compiladores mais simples podem não aceitar essa declaração). Assim, modifique o programa anterior aproveitando essa definição:

```
program exemplo7;
const comp=30;
type carac = ' '..'z';
     simb = set of carac;
var a : array[1..comp] of char;
    i,n,crsc,decr : integer;
    pal : boolean;
    pont : simb;
begin
pont:=[' ',',','.','"'];
  pal:= true;
  readln(n);
  for i:=1 to n do read(a[i]);
  crsc:=1;
  decr:=n;
  while (crsc<decr) and pal do
   if a[crsc] in pont then
   crsc:=crsc+1
   else
    if a[decr] in pont then
```

```
    decr:=decr-1
    else
      if a[crsc]=a[decr] then
      begin
      crsc:=crsc+1;
      decr:=decr-1;
      end
      else pal:=false;
  if pal then
  writeln('palindrome')
  else
  writeln('nao ha palindrome')
end.
```

Finalmente, temos a melhor estrutura para o problema do palíndromo, originalmente dividido em três etapas:

* **entrar a cadeia de caracteres**
* **verificar se ela é um palíndromo**
* **mostrar o resultado**

Cada etapa foi refinada em etapas menores até que os resultados fossem as próprias declarações em Pascal e, juntas, formassem um programa. No nosso caso, o programa era bem simples. Porem, em situações mais complexas, é aconselhável construí-lo por partes semi-independentes (*procedures*) e testá-las separadamente. Um procedure é uma sub-rotina nomeada com um rótulo e declarada junto com as variáveis. Durante a execução do programa, basta chamar o procedure através de seu rótulo, como se fosse um comando. Essa técnica permite modularizar os programas, tornando-os bem estruturados.

Há dois grupos de sistemas Pascal. Um deles usa compiladores em código específico — ou seja, escritos exclusivamente para as máquinas em que irão rodar. Como várias máquinas são similares, usando sistemas operacionais CP/M ou MS DOS, as adaptações de uma para outra sao muito simples.

O segundo grupo é conhecido como Pascal UCSD (Universidade da Califórnia, em San Diego). Nesse sistema, o Pascal é compilado para o código de um computador hipotético denominado *máquina p*, e seu código é conhecido por *código p*. Isso significa que o compilador e o mesmo para todas as máquinas. Na verdade, o UCSD é escrito em Pascal, possuindo um programa específico para cada máquina. Sua vantagem é a incrível compatibilidade. Porém, a velocidade da compilação nesse sistema é bem menor que a proporcionada pelo compilador específico.



EXPERIÊNCIAS COM O PASCAL

Se você possui um sistema Pascal para seu microcomputador, tente colocar em execução alguns dos exemplos anteriores. Para isso, depois de ter acessado o compilador Pascal, digite o programa e compile-o. Os programas devem funcionar para qualquer compilador que aceite letras minusculas.

Os microcomputadores que usam o CP/M dispõem de um compilador poderoso, o TURBO PASCAL. Para acessá-lo, basta chamar do disquete o nome TURBO, seguido de <ENTER>. Você irá obter a seguinte saudação na tela:

TURBO Pascal system Version 2.00A
CP/M-80, Z80
Copyright (C) 1983, 1984 by
BORLAND Inc.
Include error messages (Y/N)?

Pode-se optar por mensagens de erro escritas por extenso (Y) ou simplesmente descritas pelos seus códigos (N). Se você estiver iniciando seu contato com o Pascal, digite **Y** e <**ENTER**>. O sistema apresentará um menu:

Logged drive:

Work file:
Main file:

Edit Compile Run Save
eXecute Dir Quit compiler Options

Text:
Free:

Para iniciar qualquer trabalho, você terá que digitar **W** seguido do nome de um arquivo. Se ele já existir no disquete, o computador irá trazê-lo para a memória; caso contrário, criará um arquivo com esse nome.

Suponhamos que você queira criar o arquivo **exemplo7**. Tendo escrito esse nome seguido de <**ENTER**>, entre no modo de edição digitando a letra **E**.

No manual de instruções que acompanha o compilador, você encontrará orientações para movimentar o cursor por meio das teclas. Recorra a ele para escrever seu programa. Ao terminar, saia do modo de edição usando o comando apropriado (veja manual). Antes de tentar compilar o programa, guarde-o no disquete digitando **S**.

Para compilar o programa, pressione a tecla **C**. Caso haja algum erro, o compilador indicará em que posição ele ocorreu. Depois de corrigi-lo, tente compilar o programa novamente. Se o computador indicar que completou a tarefa, execute o programa com **X**.

Lembre-se de que cada declaração precisa ser seguida de um ponto e vírgula, exceto quando depois dela houver um **end**. Uma instrução composta deve vir entre um **begin** e um **end**. Se você colocar um ponto e vírgula imediatamente após um **while**, um **repeat** ou um **for**, o computador entenderá que aquele é o fim da declaração e ignorará os comandos seguintes a serem repetidos.

ORGANIZAÇÃO DE PROJETOS

- CAMINHO CRÍTICO
- REDE PERT
- ATIVIDADES, TEMPOS E EVENTOS
- CÁLCULOS

Se você deseja montar e controlar qualquer tipo de projeto — da reforma de um automóvel à compra de um imóvel —, utilize este programa para se organizar e economizar tempo.

Algumas vezes temos que realizar tarefas que incluem várias ações, cada qual exigindo um tempo diferente de execução, e todas dependendo do sucesso das etapas anteriores. Sem uma boa dose de organização e planejamento, é quase impossível saber quais etapas executar, e em que ordem. A situação fica ainda mais difícil quando há um prazo para o término da atividade.

Se calcularmos o tempo de execução de todas as ações envolvidas, veremos que há sempre uma certa sequência de eventos que determina o tempo total do projeto. A essa sequência chamamos *caminho crítico*. Qualquer atraso ou adiantamento no caminho crítico se refletirá numa alteração do tempo tomado pelo projeto. Por outro lado, uma atividade que não pertença a essa sequência especial poderá ser atrasada sem causar alterações no tempo total.

O cálculo do caminho crítico não é tão fácil se estivermos limitados ao papel e caneta, mas, com o auxílio do computador e do programa que aqui fornecemos, torna se bem mais simples e rápido. O programa permite a construção de um banco de dados contendo todas as atividades, seu tempo de execução (ou estimativas, caso não se conheça o tempo real) e a ordem em que elas deverão ser executadas. Baseado nessas informações, calcula o caminho crítico bem como o tempo livre das atividades não-críticas (aquelas que não pertencem à sequência especial). Esse último dado indica ao usuário por quanto tempo é possível interromper o projeto sem alterar seu prazo final.

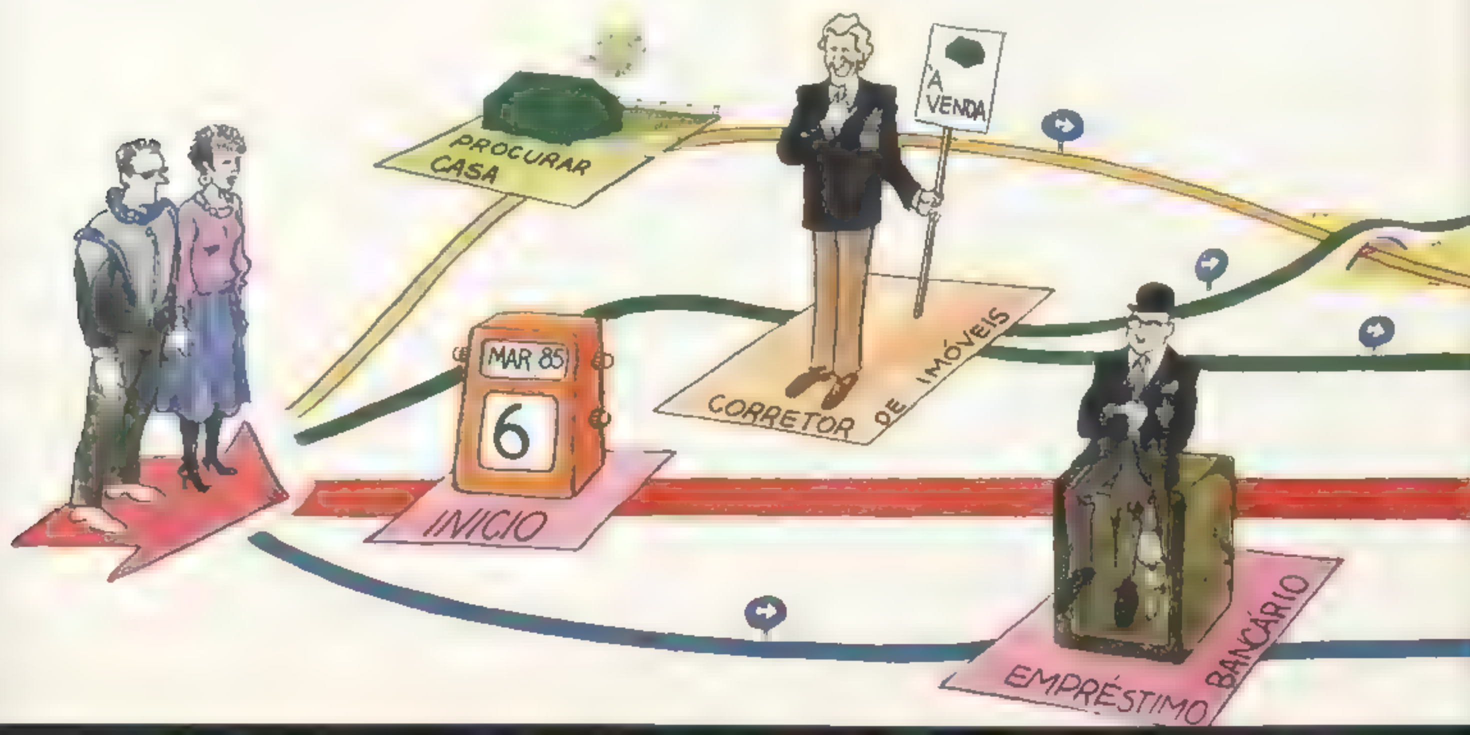
Para o cálculo do caminho crítico, utilizam-se as técnicas de programação conhecidas como CPM (do inglês *Critical Path Method*, Método do Caminho Crítico), CPA (do inglês *Critical Path Analysis*, Análise do Caminho Crítico) e PERT (do inglês *Program Evaluation and Review Technique*, Técnica de Inspeção e Estimativa de Projeto).

Não só projetos empresariais, mas qualquer projeto, por menor que seja, pode ser avaliado pelo programa.

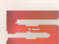
Vamos tomar como exemplo um projeto que envolve a coordenação de várias etapas diferentes, muitas vezes sob prazos rigorosos: a compra de uma casa. O diagrama da página 1452 mostra como deveria ser o fluxo PERT para a execução dessa tarefa. Os círculos assinalam os *eventos*, ou seja, momentos entre atividades cujo tempo depende de outras. Eles marcam, assim, o início ou o fim de uma atividade. As atividades estão descritas ao longo das linhas que ligam os eventos, junto com as estimativas do tempo necessário.

Até aqui, muitas atividades parecem incertas e várias linhas se cruzam. Embora o diagrama contenha toda a informação necessária, ainda é difícil saber que atividades merecem prioridade. Não se pode garantir, também, que o projeto seja concluído dentro do prazo estipulado. Além disso, à medida que avançamos no projeto de aquisição da casa, é possível que surjam outros fatores capazes de afetar as expectativas iniciais, exigindo que o diagrama seja modificado e atualizado.

Nosso programa se encarrega de todas essas questões. Na primeira parte, apresentada a seguir, cuida do banco de dados das atividades; na segunda, calcula o caminho crítico.







```
 5 BORDER 0  PAPER 4· INK 0·
 CLS
 7 POKE 23658,8  POKE 23609,
 20
10 CLS · LET false=0: LET ma
100  LET me=100· LET mh=212
LET se=-1: LET fe=-1: GOSUB
12  LET ck false· LET aa=0·
LET ee=0· GOTO 50
12 LET zz=9999: LET true=1:
LET p$="introduza ": LET a$="
 atividade"
14 DIM w$(85,32)· LET w$(.)="
Nenhuma"+a$+" PRECEDE o event
o"· LET w$(2) "EXCEDEU"
.6 LET w$(3) "VOCE NAO PODE U
SAR ESTE NUMERO "· LET w$(4)
p$+"texto para este(a) "
18 LET w$(5)=a$+"REFERE A EVE
NTO NAO DEFINIDO "
22 DEF FN a(x)=x*(x<0)·
DEF FN z(x) x*(x>0)
26 DIM a.ma)· DIM g(ma)
30 DIM w(ma)· DEF FN w(x)=ABS
x*(x<1)+ABS (2-x)*(x>1)
 DEF FN x(x)=x*(2.3/572+x*x*(
15.9402-x*x*(184 744-x*x*
688 472)))/1 20667
34 DEF FN I$(x)=(STR$ (x)+"
 ")( TO 6)
36 DEF FN b(x)=x-INT (x/256)*
256
38 DEF FN p$(x)="--"( TO INT
((6-x)/2))· DEF FN q$(x)="
"( TO INT ((6 x)/2))
40 DIM e(me)
42 DIM x(8)
44 DIM s(mh)  DIM f(mh)  DIM
u(mh): DIM t(mh)· DIM n(mh):
DIM u$(mh,20)· DIM y(mh): DIM
 z(mh)
46 DIM p(mh)· DIM q(mh)
48 DEF FN u(x)=u(ABS x+(x=0))
*(x>0)  RETURN
50 CLS : PRINT "1=define ",a$
'"2=apaga ";a$· PRINT "3=defi
ne evento"'"4=apaga evento"
60 PRINT "5=salva no gravador
"'"6=carrega do gravador"'"7=
testa gravador"'"8=mostra det
alhes"
62 PRINT "9=SAIDA"'"10=verifi
ca e ordena a rede"
64 PRINT "11=calc com tempos
medios"
66 PRINT "12=calc com incerte
zas"  PRINT
70 INPUT t· PRINT t: IF t=9
THEN  STOP
72 IF t<1 OR t>12 THEN  PRINT
 "t=";t;" NAO RECONHECIDO":
GOTO 114
74 IF t>10 AND NOT ck THEN
PRINT "FACA VERIFICACAO DE DA
DOS PRIMEIRO"· GOTO 114
76 IF aa 0 AND (t>7 OR t=5)
THEN  PRINT "IMPOSSIVEL - NAO
 FOI INTRODUZIDA NENHUMA";a$·
 GOTO 114
80 IF t>7 THEN  GOTO 100
82 IF t 6 THEN  CLEAR · LET t
=6
84 IF t>4 THEN  PRINT "Nome d
o arquivo·";· INPUT f$  PRINT
f$  GOTO 100
86 LET f$ a$· IF t>2 THEN
LET f$="evento"
88 PRINT p$;f$;" numero":
PRINT "ou zero para sair ";
90 INPUT u· PRINT u: LET u-
INT u· IF u=0 THEN  GOTO 50
92 IF u<1 OR u>zz THEN  PRINT
w$(3): GOTO 88
94 IF t>2 THEN  LET u -u
96 GOSUB 450· LET ck=false
98 IF (t=2 OR t 4) AND (0=u(x
) OR zz<u(x)) THEN  PRINT "VO
CE NUNCA USOU ESTE NUMERO":
GOTO 114
100 GOSUB 20*(t=1)+100*t+900*(
t-10)*(t>10)
112 GOTO 50
114 FOR t=1 TO 500: NEXT t:
GOTO 50
120 IF 0<u(x) AND zz> u(x)
THEN  GOSUB 942· GOSUB 932:
GOTO 130
122 IF aa=ma THEN  PRINT w$(2)
;f$: RETURN
124 LET aa=a+1: LET a(aa)=x:
LET u(x)=u
130 PRINT w$(4);f$;"·";: INPUT
u$(x): PRINT u$(x): LET xa=x
140 PRINT p$;"iniciar evento,
finalizar evento";: INPUT s,f·
PRINT s;" ";f: LET s=INT s:
LET f=INT f
142 IF s<1 OR s>zz OR f<1 OR f
>zz THEN  PRINT w$(3): GOTO
 140
150 LET u=-s: GOSUB 450: IF u(
x)<0 THEN  GOTO 156
152 IF ee=me THEN  PRINT w$(2)
;"eventos": GOTO 140
154 GOSUB 350
156 LET s(xa)=x
160 LET u=-f: GOSUB 450· IF u(
x)<0 THEN  GOTO 166
162 IF ee=me THEN  PRINT w$(2)
;"eventos": GOTO 140
164 GOSUB 350
166 LET f(xa) x
170 PRINT p$;"tempo provavel p
ara executar ";  INPUT t(xa)
PRINT t(xa)
172 IF t(xa)<0 THEN  PRINT "NA
O PODE SER FEITO COM ESTA VELO
CIDADE": GOTO 170
180 PRINT "introduza estimativ
a de tempo   com 90% de certez
a"· PRINT "Pode ser executado
em ";: INPUT n(xa)  PRINT n(xa
)
182 IF n(xa)<t(xa) THEN  PRINT
"ISTO NAO E COMPATIVEL COM O
   TEMPO PROVAVEL"  GOTO 170
190 RETURN
200 FOR b=1 TO aa· IF x=a(b)
THEN  LET a=b
220 NEXT b  LET a(a)=a(aa):
LET u(x)=zz+1· LET aa=aa 1·
RETURN
300 IF u(x)<0 THEN  GOSUB 946·
GOSUB 933· GOTO 330
310 IF ee=me THEN  PRINT w$(2)
·f$· RETURN
312 GOSUB 350
```



```
5  CLEAR
10 MA = 100:ME = 100.FA = 0 MH
= 212: GOSUB 12:CK = FA: GOTO 5
0
12 TR = 1 Z$ -  CHR$ (13) DO$ =
 Z$ +  CHR$ (4)·ZZ - 32766·QT$
=  CHR$ (34)
14 P$ = "FORNECA ":A$ = " ATIVI
DADE "
16  DIM W$(5):W$(1) = "NENHUMA"
 + A$ + " PRECEDE EVENTO":W$(2)
 = "EXCEDEU "
18 W$(3) = "NAO PODE USAR ESTE
NUMERO":W$(4) = P$ + "TEXTO PAR
A ESTE(A)"
20 W$(5) = " REFERE A EVENTO NA
O DEFINIDO "
22  DEF  FN A(X) = X * (X < 0)
 DEF  FN Z(X) = X * (X > 0)
28  DIM A(MA),G(MA)
30  DIM W(MA)· DEF  FN W(X) =
```

```
ABS (X) * (X <  = 1) +  ABS (2
- X) * (X > 1)
32  DEF  FN X(X) = X * (2.37572
 + X * X * (15.9402 - X * X * (
184 744 - X * X * 688.472))) /
1 20667
40  DIM E(ME)
44  DIM S(MH),F(MH),U(MH),T(MH)
,N(MH),US(MH),Y(MH),Z(MH)
46  DIM P(MH),Q(MH)
48  DEF  FN U(X) = U( ABS (X) +
 (X = 0)) * (X > 0)  RETURN
50  HOME : PRINT  TAB( 7);"MENU
 PRINCIPAL"· PRINT · PRINT "01
= DEFINE";A$: PRINT "02 - DELET
A";A$: PRINT "03 = DEFINE EVENT
O": PRINT "04 = DELETA EVENTO"
52  PRINT "05 = SALVA DADOS": P
RINT "06 = CARREGA DADOS"· PRIN
T "07 = DELETA ARQ DO DISCO": P
RINT "08 = MOSTRA DETALHES"
54  PRINT "09 = REINICIA"
56  PRINT "10 = VERIFICA E ORDE
NA REDE"
58  PRINT "11 = CALC COM DURACA
O MEDIA"
60  PRINT "12 = CALC COM INCERT
EZAS": PRINT "13 = SAIDA PARA "
;
62  IF KK$ = "S" THEN  INVERSE
: PRINT "IMPRESSORA": NORMAL
63  IF KK$ <  > "S" THEN  INVER
SE · PRINT "TELA": NORMAL
64  PRINT "14 = TERMINA"
66  PRINT : PRINT :T = 0: INPUT
 "SUA OPCAO (1-14) ";T
68  IF T = 14 THEN  INPUT "TEM
CERTEZA (S/N)? ";AN$· IF  LEFT$
 (AN$,1) = "S" THEN  HOME : END
69  IF T = 13 OR T = 14 THEN 10
0
72  IF T = 9 THEN 900
74  IF T < I OR T > 13 THEN  PR
INT "CODIGO",T;"NAO RECONHECIDO
"· GOTO 114
76  IF T > 10 AND  NOT (CK) THE
N  PRINT "USE OPCAO (10) PRIMEI
RO": GOTO 114
78  IF AA - 0 AND (T > 7 OR T =
 5) THEN  PRINT "IMPOSSIVEL - N
ENHUMA";A$: GOTO 114
80  IF T > 7 THEN 100
82  IF T = 6 THEN  CLEAR .DO$ =
  CHR$ (13) +  CHR$ (4)·T = 6
84  IF T > 4 THEN  HOME · PRINT
 "FORNECA NOME DO ARQ ",  INPUT
 F$  GOTO 100
86 F$ = A$: IF T > 2 THEN F$ =
" EVENTO"
88  HOME · PRINT P$;"NUMERO DO(
A)",F$· PRINT "OU ZERO PARA TER
MINAR"
90  INPUT U·U =  INT (U): IF U
= 0 THEN 50
92  IF U < 1 OR U > ZZ THEN  PR
INT W$(3): FOR E = 1 TO 1000: N
EXT : GOTO 88
94  IF T > 2 THEN U =  - U
96  GOSUB 450:CK = FA
98  IF (T = 2 OR T = 4) AND (0
= U(X) OR ZZ < U(X)) THEN  PRIN
T "VOCE NUNCA USOU ESTE NUMERO"
: GOTO 114
100  IF KK$ = "S" AND (T > 7 AN
D T < 12) THEN  PRINT DO$"PR#1"
101  HOME : ON T GOSUB 120,200,
300,400,500,600,700,800,900,100
0,2000,3000,960
105  IF KK$ = "S" AND (T > 7 AN
D T < 13) THEN  PRINT DO$"PR# 0
"
112  GOTO 50
114  FOR T = 1 TO 1000: NEXT :
GOTO 50
120  IF 0 < U(X) AND ZZ >  = U(
X) THEN  GOSUB 942: GOSUB 932:
GOTO 130
122  IF (AA = MA) THEN  PRINT W
$(2);F$· RETURN
124 AA = AA + 1:A(AA) = X:U(X)
= U
130  PRINT W$(4);F$: INPUT U$(X
):XA = X
140  PRINT P$;"EVENTO INICIAL ,
 EVENTO FINAL": INPUT S,F:S =
INT (S):F =  INT (F)
142  IF S < 1 OR S > ZZ OR F <
1 OR F > ZZ THEN  PRINT W$(3):
GOTO 140
150 U =  - S: GOSUB 450: IF U(X
) < 0 THEN 156
152  IF EE = ME THEN  PRINT W$(
2);"EVENTOS": GOTO 140
154  GOSUB 350
156 S(XA) = X
160 U =  - F. GOSUB 450: IF U(X
) < 0 THEN 166
162  IF EE = ME THEN  PRINT W$(
2);"EVENTOS": GOTO 140
164  GOSUB 350
166 F(XA) = X
170  PRINT P$,"TEMPO PROVAVEL D
E EXECUCAO": INPUT T(XA)
172  IF T(XA) < 0 THEN  PRINT "
MUITO RAPIDO !!": GOTO 170
180  PRINT "ESTIMATIVA COM 90%
DE CERTEZA": PRINT "PODE SER FE
ITO EM": INPUT N(XA)
182  IF N(XA) < T(XA) THEN  PRI
NT "NAO CONFERE COM O TEMPO PRO
VAVEL": GOTO 170
190  RETURN
200  FOR B = 1 TO AA: IF X = A(
B) THEN A = B
220  NEXT B:A(A) = A(AA):U(X) =
 ZZ + 1·AA = AA - 1: RETURN
300  IF U(X) < 0 THEN  PRINT "E
VENTOS":XP = U(X): GOSUB 950: P
RINT U$(X): GOTO 330
310  IF EE = ME THEN  PRINT W$(
2);F$: RETURN
312  GOSUB 350
330  PRINT W$(4);F$: INPUT U$(X
):S(X) = 0: RETURN
350 EE = EE + 1:E(EE) = X:S(X)
=  - 1:F(X) = 0:U(X) = U
360 T(X) = 0·N(X) = 0:US(X) = "
": RETURN
400 Z = X: FOR F = 1 TO EE: IF
E(F) = Z THEN E = F
420  NEXT F:E(E) = E(EE):U(Z) =
 ZZ + 1:EE = EE - 1: RETURN
450 Z = U -  INT ((U - 1) / MH)
 * MH:Y = 2:X = 0
460  IF X = 0 AND (0 = U(Z) OR
ZZ + 1 = U(Z)) THEN X = Z
470  IF U = U(Z) THEN X = Z: RE
TURN
480  IF Y = 1 OR U(Z) = 0 THEN
 RETURN
490 Z = Z + Y - MH *  INT ((Z +
Y - 1) / MH):Y = Y + Y - MH *
 INT ((Y + Y - 1) / MH): GOTO 4
60
500  HOME : PRINT DO$"OPEN "F$
505  PRINT DO$"DELETE "F$
510  PRINT DO$"OPEN "F$
515  PRINT DO$"WRITE "F$
520  PRINT MA;Z$;ME,Z$,MH;Z$,AA
,Z$:EE:Z$:CK
525  IF CK THEN  PRINT SE;Z$;FE
530  FOR A = 1 TO AA:X = A(A)
535  PRINT X;Z$U(X)Z$S(X)Z$F(X)
,$T(X)Z$N(X)Z$G(A)Z$US(X)
540  NEXT A
545  FOR F   1 TO EF X   E(E)
550  PRINT X·Z$U(X)Z$S(X)Z$F(X)
Z$T(X)Z$N(X)Z$US(X)
555  NEXT E
560  FOR X   1 TO MH  IF U(X)
 ZZ + 1 THEN  PRINT X
565  NEXT X  PRINT 0
570  PRINT DO$"CLOSE "F$
575  RETURN
```



```
4 MAXFILES=3
6 OPEN "CRT:" FOR OUTPUT AS#1
8 OPEN "LPT:" FOR OUTPUT AS#2
10 CLEAR 2000:MH=212:ME=100:MA=
100:FA=0:GOSUB 20:CK=FA:GOTO 14
0
20 ZZ=9999:TR=-1:P$="INTRODUZA
":A$=" ATIVIDADE":E$=CHR$(13)
30 DIMW$(5):W$(1)="NENHUMA"+A$+
" PRECEDE O EVENTO":W$(2)="EXCE
DEU"
40 W$(3)="VOCE NAO PODE USAR ES
TE NUMERO":W$(4)=P$+"TEXTO PARA
 ESTE(A) "
50 W$(5)="REFERE A EVENTO NAO D
EFINIDO"
60 DEFFNA(X)=-X*(X<0):DEFFNZ(X)
=-X*(X>0)
70 DIM A(MA),G(MA)
80 DIM W(MA):DEFFNW(X)=-ABS(X)*
(X<=1)-ABS(2-X)*(X>1)
90 DFFNX(X)=X*(2.37572+X*X*(15.
9402-X*X*(184.744-X*X*688.472))
)/1.20667
100 DIM E(ME)
110 DIM S(MH),F(MH),U(MH),T(MH)
,N(MH),US(MH),Y(MH),Z(MH)
120 DIM P(MH),Q(MH)
130 DEFFNU(X)=-U(ABS(X)-(X=0))*
(X>0):RETURN
140 CLS:PR=1:LOCATE 12,1:PRINT"
MENU PRINCIPAL":PRINT"1=DEFINE"
;A$;" OU EVENTO":PRINT"2=APAGA"
;A$;" OU EVENTO"
150 PRINT"3=SALVA DADOS":PRINT"
4=CARREGA DADOS":PRINT"5=IMPRIM
E DETALHES":PRINT"6=TERMINA"
160 PRINT"7=VERIFICA E ORDENA A
 REDE"
170 PRINT"8=CALC COM TEMPOS MED
```

```
IOS"
180 PRINT"9=CALC COM INCERTEZAS
"
190 T$=INKEY$:IF T$<"1" OR T$>"
9" THEN 190
200 T=VAL(T$):PRINT T
210 IF T>7 AND NOT(CK) THEN PRI
NT"FACA VERIFICACAO (7) PRIMEIR
O":GOTO 380
220 IF AA=0 AND (T>40 OR T=3) T
HEN PRINT"IMPOSSIVEL - NAO FOI
INTRODUZIDO";A$:GOTO380
230 IF T>4 THEN 350
240 IF T=4 THEN CLEAR 2000:T=4
250 IF T>2 THEN PRINT"NOME DO A
RQUIVO":INPUTF$:GOTO 350
260 CLS·PRINT A$;" OU EVENTO (A
/E) ?";
270 T$=INKEY$:IF T$<>"A" AND T$
<>"E" THEN 270
280 PRINT T$ IF T$="A" THEN F$=
A$ ELSE F$=" EVENTO"
290 PRINT:PRINT P$;F$;" NUMERO"
:PRINT"OU ZERO PARA SAIR"
300 INPUT U U=INT(U):IF U=0 THE
N 140
310 IF U<1 OR U>ZZ THEN PRINT W
$(3):GOTO 280
320 IF T$="E" THEN U=-U
330 GOSUB 700·CK=FA
340 IF(T=2 OT T=4)AND(0=U(X) OR
 ZZ<U(X) THEN PRINT"VOCE NUNCA
USOU ESTE NUMERO".GOTO 380
350 ON T GOSUB 390,590,750,830,
890,960,1070,1550,1660
360 IF T<3 THEN 290
370 GOTO 140
380 FOR T=1 TO 1000·NEXT T GOTO
 140
390 IF F$<>A$ THEN 620
400 IF 0<U(X) AND ZZ>=U(X) GOSU
B 1030:GOSUB 1000:GOTO 430
410 IF AA=MA THEN PRINTW$(2);F$
·RETURN
420 AA=AA+1:A(AA)=X:U(X)=U
430 PRINT W$(4);F$ INPUT U$(X)
XA=X
440 PRINT P$;"INICIAR EVENTO, F
INALIZAR EVENTO":INPUT S,F:S=IN
T(S)·F=INT(F)
450 IF S<1 OR S>ZZ OR F<1 OR F>
ZZ THEN PRINT W$(3)·GOTO 440
460 U=-S:GOSUB 700:IF U(X)<0 TH
EN 490
470 IF EE=ME THEN PRINT W$(2),"
EVENTOS":GOTO 440
480 GOSUB 660
490 S(XA)=X
500 U=-F:GOSUB 700.IF U(X)<0 TH
EN 530
510 IF EE=ME THEN PRINT W$(2);"
EVENTOS":GOTO 440
520 GOSUB 660
530 F(XA)=X
540 PRINT P$:"TEMPO PROVAVEL PA
RA EXECUTAR":INPUT T(XA)
550 IF T(XA)<0 THEN PRINT"NAO P
ODE SER  FEITO  COM  ESTA  VELO
CIDADE" GOTO 540
560 PRINT"INTRODUZA ESTIMATIVA
DE TEMPO   COM 90% DE CERTEZA":
PRINT"PODE SER FEITO EM":INPUT
N(XA)
570 IF N(XA)<T(XA) THEN PRINT "
NAO E COMPATIVEL COM O TEMPO
 PROVAVEL"·GOTO 540
580 RETURN
```



```
10 PCLEAR 1:CLEAR 2000 MH=212:M
E=100:MA=100.FA=0.GOSUB 20.CK=F
A·GOTO 140
20 ZZ=9999:TR=-1:P$="INTRODUZA
":A$=" ATIVIDADE" E$=CHR$(13)
30 DIMW$(5)·W$(1)="NENHUMA"+A$+
" PRECEDE O EVENTO"·W$(2)="EXCE
DEU"
40 W$(3)="VOCE NAO PODE USAR ES
TE NUMERO" W$(4)=P$+"TEXTO PARA
 ESTE(A) "
50 W$(5)="REFERE A EVENTO NAO D
EFINIDO"
60 DEFFNA(X)= X*(X<0):DEFFNZ(X)
= X*(X>0)
70 DIM A(MA),G(MA)
80 DIM W(MA)·DEFFNW(X)=-ABS(X)*
(X<=1)-ABS(2-X)*(X>1)
90 DEFFNX(X)=X*(2 37572+X*X*(15.
9402-X*X*(184.744-X*X*688 472))
)/1.20667
100 DIM E(ME)
110 DIM S(MH),F(MH),U(MH).T(MH)
,N(MH),U$(MH),Y(MH),Z(MH)
120 DIM P(MH),Q(MH)
130 DEFFNU(X)= U(ABS(X) (X=0))*
(X>0):RETURN
140 CLS PR=0·PRINT @8,"MENU PRI
NCIPAL"·PRINT"1=DEFINE",A$," OU
 EVENTO":PRINT"2=APAGA";A$;" OU
 EVENTO"
150 PRINT"3=SALVA DADOS":PRINT"
4=CARREGA DADOS" PRINT"5=IMPRIM
E DETALHES" PRINT"6=SAIDA"
160 PRINT"7=VERIFICA E ORDENA A
 REDE"
170 PRINT"8=CALC COM TEMPOS MED
IOS"
180 PRINT"9=CALC COM INCERTEZAS
":PRINT"?";
190 T$=INKEY$·IF T$<"1" OR T$>"
9" THEN 190
200 T=VAL(T$):PRINT T
210 IF T>7 AND NOT(CK) THEN PRI
NT"FACA VERIFICACAO (7) PRIMEIR
O":GOTO 380
220 IF AA=0 AND (T>40 OR T=3) T
HEN PRINT"IMPOSSIVEL - NAO FOI
INTRODUZIDO".A$:GOTO380
230 IF T>4 THEN 350
240 IF T=4 THEN CLEAR 2000:T=4
250 IF T>2 THEN PRINT"NOME DO A
RQUIVO"·INPUTF$·GOTO 350
260 CLS:PRINT A$;" OU EVENTO (A
/E) ?";
270 T$=INKEY$:IF T$<>"A" AND T$
<>"E" THEN 270
280 PRINT T$ IF T$="A" THEN F$=
A$ ELSE F$=" EVENTO"
290 PRINT PRINT P$.F$." NUMERO"
 PRINT"OU ZERO PARA SAIR"
300 INPUT U U=INT(U)·IF U=0 THE
N 140
310 IF U<1 OR U>ZZ THEN PRINT W
$(3) GOTO 280
320 IF T$="E" THEN U=-U
330 GOSUB 700 CK=FA
340 IF(T=2 OT T=4)AND(0=U(X) OR
 ZZ<U(X) THEN PRINT"VOCE NUNCA
USOU ESTE NUMERO" GOTO 380
350 ON T GOSUB 390,590,750,830
890,960,1070,1550,1660
360 IF T<3 THEN 290
370 GOTO 140
380 FOR T=1 TO 1000 NEXT T GOTO
 140
390 IF F$<>A$ THEN 620
400 IF 0<U(X) AND ZZ>=U(X) GOSU
B 1030 GOSUB 1000 GOTO 430
410 IF AA=MA THEN PRINTW$(2):F$
 RETURN
420 AA=AA+1:A(AA)=X U(X)=U
430 PRINT W$(4):F$·INPUT U$(X)
XA=X
440 PRINT P$."INICIAR EVENTO, F
INALIZAR EVENTO".INPUT S.F S=IN
T(S) F=INT(F)
450 IF S<1 OR S>ZZ OR F<1 OR F>
ZZ THEN PRINT W$(3) GOTO 440
460 U=-S:GOSUB 700:IF U(X)<0 TH
EN 490
470 IF EE=ME THEN PRINT W$(2) "
EVENTOS" GOTO 440
480 GOSUB 660
490 S(XA)=X
500 U=-F:GOSUB 700:IF U(X)<0 TH
EN 530
510 IF EE=ME THEN PRINT W$(2):"
EVENTOS":GOTO 440
520 GOSUB 660
530 F(XA)=X
540 PRINT P$;"TEMPO PROVAVEL PA
RA EXECUTAR":INPUT T(XA)
550 IF T(XA)<0 THEN PRINT"NAO P
ODE SER  FEITO  COM  ESTA  VELO
CIDADE":GOTO 540
560 PRINT"INTRODUZA ESTIMATIVA
DE TEMPO   COM 90% DE CERTEZA"·
PRINT"PODE SER FEITO EM":INPUT
N(XA)
570 IF N(XA)<T(XA) THEN PRINT "
NAO E COMPATIVEL COM O TEMPO
 PROVAVEL":GOTO 540
580 RETURN
```

PLANEJAMENTO DA REDE PERT

Antes de usar o programa para avaliar um projeto, você terá que dividi-lo em atividades individuais e estimar o tempo de execução de cada uma delas. Para isso, o mais conveniente, como já vimos, é desenhar um esquema de blocos: a rede PERT.

À medida que for desenhando o diagrama, descreva as atividades ao longo das linhas de conexão entre eventos. Dentro dos círculos, coloque a descrição dos eventos. Se estes forem apenas passos intermediários entre atividades, não é necessário atribuir-lhes nomes específicos.

A versão do programa para o Spectrum permite o uso de até vinte caracte-

res por evento ou atividade; nos demais micros, podem-se utilizar 255 caracteres. Entretanto, dê preferência a títulos sucintos, para não sobrecarregar a memória disponível.

Se você tem uma noção mais ou menos precisa da duração de cada atividade, escreva esse valor no diagrama. Posteriormente, veremos como estimar os tempos. Lembre-se, porém, de que a unidade de medida de tempo escolhida (hora, dia, semana etc.) deve ser sempre a mesma em toda a rede.

O programa só funcionará corretamente se a rede for viável do ponto de vista lógico — deve ter apenas um ponto de partida e um de chegada, e não formar laços ou caminhos circulares.

O passo seguinte ao desenho da rede PERT é a numeração de todas as atividades e eventos e sua introdução no programa. A ordem dos números não é importante, mas o computador precisa deles para trabalhar. Um método bastante simples consiste em numerar os eventos de dez em dez, como se faz com um programa em BASIC — o que permite inserir números extras entre os já existentes, se for necessário.

Quando você for definir os eventos e as atividades, o computador pedirá o número e a descrição de cada um deles, além de uma estimativa do seu tempo médio e do seu tempo pessimista (aquele que conta com 90% de possibilidade de não ser ultrapassado).

Na vida real, raramente podemos calcular com precisão o tempo que levaremos para realizar uma atividade — mesmo que a tenhamos executado muitas vezes. Entretanto, não é difícil estimar sua duração média, bem como sua duração aproximada. Isso é tudo o que o programa exige em termos de estimativa de tempo. No entanto, como você verá, os cálculos que ele faz não são tão simples, pois devem levar em conta quatro diferentes situações.

A primeira situação é aquela em que estamos absolutamente seguros quanto ao tempo que uma certa atividade levará. Por exemplo, se as instruções para a construção de uma casa recomendam que deixemos o cimento secar por 48 horas, é exatamente isso que vamos fazer! Nesse caso, ao usar o programa, devemos indicar o mesmo valor para o tempo médio e o pessimista.

A segunda situação ocorre quando estamos razoavelmente seguros sobre a duração de determinada atividade. Sabemos, por exemplo, que uma viagem entre as cidades de São Paulo e Campinas, de automóvel, demora cerca de 90 minutos, porque já a repetimos muitas vezes, sem observar grandes variações (mais ou menos 10%, devido a eventuais dificuldades de trânsito). Portanto, entramos 90 minutos como tempo médio, e 100 minutos como tempo pessimista. Essa situação corresponde à curva conhecida como curva *normal* ou *gaussiana*.

A terceira situação é a que chamamos "espere até acontecer". Por exemplo você só terá a certeza de que a reforma no telhado foi bem feita quando começar a chover. Aqui, o tempo pessimista é cerca de 2,5 vezes maior que o tempo médio. Este pode ser obtido a partir do número de dias de chuva naquele mês específico.

A quarta e última situação corresponde ao tempo do "tudo ou nada". É muito improvável, por exemplo, que uma determinada peça do carro quebre, mas, se isso ocorrer, o veículo ficará na oficina durante dez dias. Nesse caso, tome o valor máximo (dez dias) como o tempo pessimista, e a média aritmética (10 vezes 1/100, ou 1/10 de dia) como o tempo médio.

Não é necessário indicar ao microcomputador o gráfico em que se encaixa cada atividade. A máquina simplesmente toma as duas estimativas de tempo, e efetua os cálculos. Se a diferença entre os dois valores for de 0 a 30%, ele usa a curva gaussiana; se for de 30 a 130%, usa uma curva gaussiana modificada. Por outro lado, se o valor da media estiver entre 130 e 300%, o programa recorre à curva exponencial e, se estiver acima dessa porcentagem, utiliza a curva bimodal.

Depois de entrar todas as atividades e suas respectivas estimativas de tempo, você deverá entrar os eventos. Para isso, basta digitar o número e a descrição de cada evento no diagrama. Se você cometer algum erro, chame as opções de apagamento ou de modificação de eventos ou atividades.

A informação entrada no computador pode ser exibida em vários tipos de tabela. A opção "Mostre Detalhes" mostrará uma lista de tudo o que foi entrado. Se você tem uma impressora, poderá também obter uma cópia em papel.

CONSISTÊNCIA DE DADOS

Antes que o computador prossiga com os cálculos, é necessário checar se há consistência lógica na rede entrada. Se esta contiver caminhos circulares, o programa cairá em uma alça sem fim de cálculos e, eventualmente, apresentará um defeito de execução.

Escolha a opção de verificação e observe o resultado. Se tudo estiver bem, o programa imprimirá os números dos eventos de início e de fim. Entretanto, se encontrar alguma inconsistência na rede, emitirá uma mensagem de erro, identificando caminhos circulares ou interrupções.

O CAMINHO CRÍTICO

Finalmente, selecione a opção para calcular o caminho crítico. Existem duas alternativas: a primeira usa o tempo médio de cada atividade, e a segunda, o tempo incerto (pessimista). Peça antes a primeira alternativa.

O vídeo mostrará todas as atividades, seu número de código e descrição. Em seguida, dirá quando cada uma deve ser iniciada e concluída, se existe alguma folga, e se a atividade é crítica — ou se ja, se está no caminho crítico, com folga igual a zero.

O tempo é expresso na mesma unidade que foi entrado. Assim, se você usou dias, todos os tempos irão se referir a dias, contados a partir do evento inicial da rede. Suponhamos que a tabela indique que a atividade 3 deve começar no dia 6, terminar no dia 10, e tem uma folga de dois dias. Você poderá começar aquela atividade no mínimo seis dias depois de iniciado o projeto, com uma tolerância de dois dias (ou seja, se você atrasar a atividade até o dia 8, o tempo total do projeto não será alterado).

As atividades que têm folga zero devem começar exatamente no dia indicado; caso contrário, haverá um atraso no projeto. Se isso ocorrer, você precisará recalcular toda a rede, com a nova estimativa para a atividade ou atividades infratoras.

Se a maioria dos tempos que você entra são incertos, e o tempo pessimista é muito distinto do médio, use a segunda opção de cálculo, pois o caminho crítico será diferente

Quando você escolhe essa opção, o computador considera cada atividade e, usando o gráfico de distribuição mais apropriado, seleciona um tempo ao acaso, dentro dos limites propostos. A partir desse tempo, ele calcula o caminho crítico para toda a rede, exatamente como na opção anterior, e armazena os valores de início e de folga das atividades. O processo se repete 44 vezes: em cada uma delas um novo número aleatório é escolhido para as atividades incertas. As 45 repetições (todas são exibidas na tela) proporcionam uma amostra razoável para a derivação dos valores médios de tempo.

Os tempos de início e fim do projeto correspondem à média de 45 repetições

ao acaso; portanto, são bastante confiáveis. A folga média das atividades não críticas também é indicada.

Os valores críticos mostram a porcentagem da participação de certa atividade no caminho crítico — 100% significa que a atividade será sempre crítica; 0%, que ela nunca é crítica.

O último valor mostra o desvio padrão do tempo de folga, indicando a variação possível da folga e, também, a confiabilidade da estimativa. Por exemplo, se há uma folga de 1,5 dia e o desvio é igual a 1, a folga pode variar entre 0,5 dia e 2,5 dias. Nesse caso, conclui-se que a folga é pouco confiável. Ao contrário, um desvio padrão de 0,1 mostra que o valor atribuído à folga é bastante confiável.

S

```
330 PRINT w$(4);f$;"·";· INPUT
 u$(x)  PRINT u$(x)  LET s(x)
0  RETURN
350 LET ee=ee+1. LET e(ee)=x
LET s(x)=-1  LET f(x) 0  LET u
(x)=u
360 LET t(x)=0  LET n(x)=0:
LET u$(x) ""  RETURN
400 LET z=x  FOR f 1 TO ee· IF
e(f)=z THEN  LET e f
420 NEXT f  LE  e(e)=e(ee)·
LET u(z)=zz+1  LET ee=ee-1
RETURN
450 LET z u INT ((u-1)/mh)*mh
LET y 2  LET x 0
460 IF x=0 THEN  IF 0=u(z) OR
zz+1 u(z) THEN  LET x z
470 IF u u(z) THEN  LET x=z
RETURN
480 IF y=1 OR 0=u(x  THEN
RETURN
490 LET z=z+y mh*INT ((z+y-1)/
mh)  LET y y+y-mh*INT ((y+y 1)
/mh)  GOTO 460
500 LET x(1)=ma  LET x(2)=me
LET x(3) mh  LET x(4) aa  LET
x(5) ee  LET x(6)=ck  LET x(7)
=se  LET x(8)=fe  PRINT "press
ione ENTER dez vezes"  SAVE
f$+"x" DATA x()  FOR x=1 TO
10  NEXT x
510 SAVE f$+"a" DATA a()· SAVE
f$+"e" DATA e()· SAVE f$+"f"
DATA f()  SAVE f$+"g" DATA g()
  SAVE f$+"n" DATA n()  SAVE
f$+"s" DATA s()  SAVE f$+"t"
DATA t()  SAVE f$+" " DATA u()
  SAVE f$+"uS" DATA u$()·
 RETURN
600 LOAD f$+"x" DATA x()  LET
ma x 1  LET me=x(2  LET mh x
(3)  LET aa x(4)  LET ee x( )
LET ck x(6)  LET se x(7)  LET
fe=x(8)· GOSUB 12
610 LOAD f$+"a" DATA a()  LOAD
f$+"e" DATA e()· LOAD f$+"f"
DATA f()  LOAD f$+"g" DATA g()
 LOAD f$+"n" DATA n(): LOAD
f$+"s" DATA s()  LOAD f$+"t"
DATA t()  LOAD f$+"u" DATA u()
: LOAD f$+"u$" DATA u$(): LET
false 0· RETURN
 700 LET x(1)=ma: LET x(2)=me:
LET x(3)=mh: LET x(4)=aa: LET
x(5)=ee· LET x(6)=ck· LET x(7)
=se: LET x(8)=fe: VERIFY f$+"x
" DATA x()
 710 VERIFY f$+"a" DATA a()
VERIFY f$+"e" DATA e()  VERIFY
f$+"f" DATA f()· VERIFY f$+"g"
DATA g()  VERIFY f$+"n" DATA n
()  VERIFY f$+"s" DATA s()
VERIFY f$+"t" DATA t()· VERIFY
f$+"u" DATA u(): VERIFY f$+"u$
" DATA u$(): RETURN
 800 GOSUB 942
 810 FOR a=1 TO aa· LET x=a(a)
GOSUB 932
 820 LET y y+1+(LEN u$(x)>4)
IF y>20 AND a<aa THEN  GOSUB
 940  GOSUB 942
 830 NEXT a· GOSUB 940
 840 GOSUB 946· FOR e=1 TO ee
LET x=e(e)  GOSUB 933
 850 LET y y+ +(LEN u$(x)>4)
IF y 20 AND e ee THEN  GOSUB
940: GOSUB 946
 860 NEXT e  GOTO 940
 932 PRINT FN l$(FN u(s(x))),FN
l$(FN u(f(x)));FN l$(t(x));FN
l$(n(x)),ABS u(x);" ",u$(x)
RETURN
 933 PRINT ABS u(x),u$(x):
RETURN
 940 PRINT "pressione <ENTER> p
ara sair"  INPUT f$· CLS ·
RETURN
 942 CLS   PRINT "INICI FINAL T
EMPO 90% CODIGO TEXTO"
 944 PRINT "FINAL INICI PROVL E
STIM "  LET y 3· RETURN
 946 PRINT "CODIGO","TEXTO":
RETURN
 948 PRINT "PREV FINAL MIN  MAX
": LET y=3: RETURN
1000 LET ck true· FOR a=1 TO aa
  LET x=a(a)
1020 LET z=s(x): IF s(z)<0 OR z
z<u(z) THEN  PRINT u(x);w$(5);u
(z)  LET ck=false
1030 LET z=f(x)· IF s(z)<0 OR z
z<u(z) THEN  PRINT u(x);w$(5);u
(z): LET ck=false
1040 NEXT a. IF ck=false THEN
GOTO 1750
1050 LET e=1
1060 LET z=e(e)  IF s(z)<0 THEN
  GOSUB 400  IF e<=ee THEN  GOT
O 1060
1070 LET e=e+1: IF e<=ee THEN
GOTO 1060
1080 FOR e=1 TO ee· LET z=e(e):
 LET s(z)=0: LET f(z)=0: NEXT e
1082 FOR a=1 TO aa· LET x=a(a)·
 LET s(f(x))=x: NEXT a
1090 LET se=0  FOR e=1 TO ee· L
ET z=e(e)  IF s(z)>0 THEN  GOTO
 1096
1092 IF se=0 THEN  LET se=z: GO
TO 1096
1094 PRINT w$(1),u(z)  IF se<=m
h THEN  PRINT w$(1);u(se): LET
se=mh+1
1096 NEXT e· IF se 0 THEN  PRIN
T "TODOS OS EVENTOS TEM",a$:" P
RECEDENDO"
1098 IF se 0 OR se>mh THEN  GOT
O 1750
1100 FOR e=1 TO ee  LET z=e(e):
 LET t(z)=0: LET n(z)=0: NEXT e
: LET t(se)=1
1110 LET last=1: FOR c=2 TO ee+
2  IF last<>c 1 THEN  GOTO 1170
1120 FOR a=1 TO aa· LET x=a(a)·
 LET y=s(x): IF t(y)<>c-1 THEN
 GOTO 1160
1130 IF y=f(x) THEN  GOSUB 1200
: GOTO 1160
1140 IF y<>se THEN  LET y s(y)
 GOTO 1130
1150 LET y=f(x): LET s(y)=s(x):
 LET f(s(y))=y: LET t(y)=c: LET
 fe=y: LET last=c
1160 NEXT a
1170 NEXT c· PRINT "evento inic
ial ";u(se);"evento final ",u(f
e)
1180 FOR e=1 TO ee: LET y=e(e)
1190 IF f(y)=0 AND y<>fe THEN
PRINT u(y);"NAO CONECTADO AO EV
ENTO FINAL": LET ck=false
1192 NEXT e: IF ck THEN  GOTO 1
300
1194 GOTO 1750
1200 CLS : PRINT "EXISTE O SEGU
INTE LOOP": PRINT "EVENTOS...":
 LET xa=a(a)
1210 LET x=f(xa)· PRINT u(x)· L
ET y=s(xa): PRINT u(y)
1220 LET y=s(y)  PRINT u(y)  IF
 y<>x THEN  GOTO 1220
1230 RETURN
1300 LET k=1: LET ak=aa: IF aa=
1 THEN  LET k=0
1310 LET ak=INT ((ak+k)/2): IF
ak=0 THEN  GOTO 1500
1320 LET k 0  FOR a ak+1 TO aa
 LET b=a ak  LET x a(a)  LET y=
a(b)  LET xe s(x)  LET ye s y)
1330 IF t(ye)+ye/zz  t(xe)+xe/z
z THEN  GOTO 1360
1340 LET a(a)=y: LET a(b)=x: LE
T k=1
1360 NEXT a  GOTO 1310
1500 LET n(fe) last  FOR d last
-1 TO 1 STEP  1
1520 FOR a 1 TO aa  LET x a(a)
 IF n(t x),  d+1 THEN  GOTO 156
0
1550 LET y s(x  LET t(y)=1 <)
 LET n(y) 1
1560 NEXT a  NEXT d
1600 FOR a=1 TO aa: LET g(a)=a(
a): NEXT a: LET k=1: LET ak=aa:
 IF aa=1 THEN  LET k 0
1610 LET ak  NT ((ak+k)/2)  IF
ak 0 THEN  GOTO 1700
1620 LET k 0  FOR a ak+  TO aa
 LET b b-ak  LET x g(a)  LET y=
g(b)  LET xe f(x)  LET ye=f(y)
1630 IF n(ye)+ye/zz  n(xe +xe/z
z THEN  GOTO 1660
1640 LET g(a) y  LET g b)=x  LE
```

```
T k=1
1660 NEXT a: GOTO 1610
1700 LET ck=true: RETURN
1750 LET ck=false: PRINT AT 21,8,"QUALQUER TECLA PARA CONTINUAR": PAUSE 0: RETURN
2000 FOR a=1 TO aa: LET x=a(a): LET z(x)=l(x): NEXT a: GOSUB 2100
2020 FOR a=1 TO aa: LET x=a(a): LET y(x)=(z(f(x))-y(s(x))-z(x))*100: NEXT a
2030 FOR b=1 TO aa STEP 5: CLS : FOR a=b TO aa+FN a(b+4,aa): LET x=a(a)
2040 PRINT a$;" ",u(x)," ";u$(x)( TO 16)
2050 LET c=y(s(x)): LET d=z(f(x)): PRINT "pode iniciar ";c,".deve terminar ";d
2060 PRINT "tempo livre ";d-c-z(x);" (critico ",y(x);"%)": IF t=12 THEN PRINT "desvio=";q(x)
:
2070 PRINT : NEXT a: GOSUB 940: NEXT b: RETURN
2100 FOR e=1 TO ee: LET y(e(e))=0: NEXT e
2110 FOR a=1 TO aa: LET x=a(a): LET y(f(x))=y(f(x))+FN z(y(f(x)),y(s(x))+z(x)): NEXT a
2120 FOR e=1 TO ee: LET z(e(e))=y(te): NEXT e: FOR a=aa TO 1 STEP -1: LET x=a(a)
2130 LET z(s(x))=z(s(x))+FN a(z(f(x))-z(s(x))-z(x)): NEXT a: RETURN
3000 FOR a=1 TO aa: LET x=a(a): LET p(x)=0: LET q(x)=0: LET y(x)=0: NEXT a
3020 FOR e=1 TO ee: LET z=e(e): LET p(z)=0: LET q(z)=0: NEXT e
3030 FOR m=1 TO 43 STEP 3: FOR a=1 TO aa: LET w(a)=2*RND-1: NEXT a
3040 FOR n=0 TO 4 STEP 2: CLS : PRINT "CASO # ";m+n/2," EM 45"
3050 FOR a=1 TO aa: LET x=a(a): LET tx=l(x): IF tx=0 THEN LET z(x)=0: GOTO 3080
3052 LET nx=n(x): IF nx=tx THEN LET z(x)=tx: GOTO 3080
3054 LET w=FN w(w(a)+n/3): IF nx<tx*3 THEN LET z(x)=n(x)^(w*tx/nx): GOTO 3080
3060 IF nx<tx*2.34 THEN LET z(x)=tx*LN w: GOTO 3080
3070 LET w=FN x(w,5): LET z(x)=ABS (tx+w*(nx-tx))
3080 NEXT a
3090 GOSUB 2100
3100 FOR a=1 TO aa: LET x=a(a): LET z=z(f(x))-y(s(x))-z(s)
3110 LET p(x)=p(x)+z: LET q(x)=q(x)+z*z: LET y(x)=y(x)+(z<1e-6): NEXT a
3120 FOR e=1 TO ee: LET z=e(e): LET p(z)=p(z)+y(z): LET q(z)=q(z)+z(z): NEXT e: NEXT n: NEXT m
3200 FOR e=1 TO ee: LET z=e(e): LET y(z)=VAL (FN I$(p(z)/45))
3210 LET z(z)=VAL (FN I$(q(z)/45)): NEXT e
3220 FOR a=1 TO aa: LET x=a(a): LET y=y(x): LET y(x)=VAL ((STR$ (y/45*100)+"    ")( TO 4))
3230 IF p(x)<1e-2 THEN LET p(x)=0
3240 LET z=(45-y)+1e-9: LET z(x)=z(f(x))-y(s(x))-VAL (FN I$(p(x)/z))
3250 LET q(x)=SQR ABS ((q(x)-p(x)*p(x)/z)/((z-1)+1e-9)): IF q(x)<1e-6 THEN LET q(x)=0
3260 NEXT a: GOTO 2030
```



```
590 IF F$<>A$ THEN 680
600 FOR B=1 TO AA:IF X=A(B) THEN A=B
610 NEXT B:A(A)=A(AA):U(X)=ZZ+1:AA=AA-1:RETURN
620 IF U(X)<0 GOSUB 1050:PRINT USING("#####  ";ABS(U(X)),:PRINT US(X):GOTO 650
630 IF EE=ME THEN PRINT W$(2);F$:RETURN
640 GOSUB 660
650 PRINT W$(4);F$:INPUT U$(X):S(X)=0:RETURN
660 EE=EE+1:E(EE)=X:S(X)=-1:F(X)=0:U(X)=U
670 T(X)=0:N(X)=0:U$(X)="":RETURN
680 Z=X:FOR F=1 TO EE:IF E(F)=Z THEN E=F
690 NEXT F:E(E)=E(EE):U(Z)=ZZ+1:EE=EE-1:RETURN
700 Z=U-INT((U-1)/MH)*MH:Y=2:X=0
710 IF X=0 (0=U(Z) OR ZZ+1=U(Z))THEN X=Z
720 IF U=U(Z) THEN X=Z:RETURN
730 IF Y=1 OR 0=U(Z) THEN RETURN
740 Z=Z+Y-MH*INT((Z+Y-1)/MH):Y=Y+Y-MH*INT((Y+Y-1)/MH):GOTO 170
750 OPEN "O",#-1,F$:PRINT #-1,MA,ME,MH;AA;EE,CK
760 IF CK THEN PRINT#-1,SE,FE
770 FOR A=1 TO AA:X=A(A):PRINT #-1,X,U(X);S(X);F(X),T(X),N(X);G(A);U$(X):NEXT A
780 FOR E=1 TO EE:Z=E(E):PRINT#-1,Z,U(Z);S(Z);F(Z);T(Z);N(Z);U$(Z):NEXT E
790 FOR X=1 TO MH:IF U(X)=ZZ+1 THEN PRINT #-1,X
800 NEXT X:PRINT# 1,0
810 CLOSE#-1:MOTORON:FOR X=1 TO 100:NEXT X:MOTOROFF:RETURN
820 CLS:PRINT"SALVAR DADOS - erro:FOR K=1 TO 1000:NEXT:RETURN
830 OPEN"I",#-1,F$:INPUT#-1,MA,ME,MH,AA,EE,CK:GOSUB 20
840 IF CK THEN INPUT #-1,SE,FE
850 FOR A=1 TO AA:INPUT #-1,X,U(X),S(X),F(X),T(X),N(X),G(A),U$(X):A(A)=X:NEXT A
860 FOR E=1 TO EE:INPUT#-1,Z,U(Z),S(Z),F(Z),T(Z),N(Z),U$(Z):E(E)=Z:NEXT E
870 INPUT #-1,X:IF X>0 THEN U(X)=ZZ+1:GOTO 870
880 CLOSE #-1:RETURN
890 GOSUB 1870:A=0:GOSUB 1030
900 FOR A=1 TO AA:X=A(A):GOSUB 1000
910 Y=Y+1:IF Y>8 AND A<AA GOSUB 1020:GOSUB 1030
920 NEXT A:GOSUB 1020
930 E=0:GOSUB 1050:FOR E=1 TO EE:X=E(E):PRINT #PR,USING"#####   ";ABS(U(X));:PRINT #PR,U$(X)
940 Y=Y+1:IF Y>15 AND E<EE GOSUB 1020:GOSUB 1050
950 NEXT E:GOTO 1020
960 CLS:PRINT"TEM CERTEZA ?  (S/N)"
970 T$=INKEY$:IF T$<>"S" AND T$<>"N" THEN 970
980 IF T$="N" THEN RETURN
990 CLS:END
1000 PRINT #PR,USING"##### ##### #####  #####  #####";FNU(S(X)),FNU(F(X)),T(X),N(X),ABS(U(X)),:IF PR=0 THEN PRINT
1010 PRINT #PR," ";U$(X):RETURN
1020 IF PR=0 THEN PRINT"<ENTER> PARA CONTINUAR":INPUT F$:CLS:RETURN ELSE RETURN
1030 CLS:IF PR=0 OR A=0 THEN PRINT #PR,"INICIO ULTIMA VEZ   90% ATIVIDADE":Y=3
1040 RETURN
1050 CLS:IF PR=0 OR E=0 THEN PRINT #PR,"  EVENTO  TEXTO":Y=3
1060 RETURN
1070 CK=TR:FOR A=1 TO AA:X=A(A)
1080 Z=S(X):IF S(Z)<0 OR ZZ<U(Z) THEN PRINT A$;U(X);W$(5);U(Z):CK=FA
1090 Z=F(X):IF S(Z)<0 OR ZZ<U(Z) THEN PRINT A$,U(X),W$(5);U(Z):CK=FA
1100 NEXT A:IF CK=FA THEN 1540
1110 E=1
1120 Z=E(E):IF S(Z)<0 GOSUB 680:IF E<=EE THEN 1120
1130 E=E+1:IF E<=EE THEN 1120
1140 FOR E=1 TO EE:Z=E(E):S(Z)=0:F(Z)=0:NEXT E
1150 FOR A=1 TO AA:X=A(A):S(F(X))=X:NEXT A
1160 SE=0:FOR E=1 TO EE:Z=E(E):IF S(Z)>0 THEN 1190
1170 IF SE=0 THEN SE=Z:GOTO 1190
1180 PRINT W$(1);U(Z):IF SE<=MH THEN PRINT W$(1),U(SE):SE=MH+1
1190 NEXT E:IF SE=0 THEN PRINT"TODOS OS EVENTOS TEM":PRINT A$;"PRECEDENDO"
1200 IF SE=0 OR SE>MH THEN 1540
1210 FOR E=1 TO EE:Z=E(E):T(Z)=0:N(Z)=0:NEXT E:T(SE)=1
1220 LA=1:FOR C=2 TO EE+2:IF LA<>C-1 THEN 1280
1230 FOR A=1 TO AA:X=A(A):Y=S(X):IF T(Y)<>C-1 THEN 1270
1240 IF Y=F(X) GOSUB 1330:GOTO 1270
1250 IF Y<>SE THEN Y=S(Y):GOTO 1240
1260 Y=F(X):S(Y)=S(X):F(S(Y))=Y
```

```
:T(Y)=C:FE=Y:LA=C
1270 NEXT A
1280 NEXT C:PRINT"EVENTO INICIAL",U(SE);", EVENTO FINAL ",U(FE)
1290 FOR E=1 TO EE:Y=E(E)
1300 IF F(Y)=0 AND Y<>FE THEN PRINT U(Y),"NAO CONECTADO AO EVENTO FINAL":CK=FA
1310 NEXT E:IF CK THEN 1370
1320 GOTO 1540
1330 CLS:CK=FA:PRINT"EXISTE O SEGUINTE LOOP":PRINT"EVENTOS ...":XA=A(A)
1340 X=F(XA):PRINT U(X):Y=S(XA):PRINT U(Y)
1350 Y=S(Y):PRINT U(Y):IF Y<>X THEN 1350
1360 FOR X=1 TO 1000:NEXT:RETURN
1370 K=1:AK=AA:IF AA=1 THEN K=0
1380 AK=INT((AK+K)/2):IF AK=0 THEN 1430
1390 K=0:FOR A=AK+1 TO AA:B=A-AK:X=A(A):Y=A(B):XE=S(X):YE=S(Y)
1400 IF T(YE)+YE/ZZ<=XE/ZZ+T(XE) THEN 1420
1410 A(A)=Y:A(B)=X:K=1
1420 NEXT A:GOTO 1380
1430 N(FE)=LA:FOR D=LA-1 TO 1 STEP-1
1440 FOR A=1 TO AA:X=A(A):IF N(F(X))<>D+1 THEN 1460
1450 Y=S(X):F(Y)=F(X):N(Y)=D
1460 NEXT A:NEXT D
1470 FOR A=1 TO AA:G(A)=A(A):NEXT A:K=1:AK=AA:IF AA=1 THEN K=0
1480 AK=INT((AK+K)/2):IF AK=0 THEN 1530
1490 K=0: FOR A=AK+1 TO AA:B=A-AK:X=G(A):Y=G(B):XE=F(X):YE=F(Y)
1500 IF N(YE)+YE/ZZ<=XE/ZZ+N(XE) THEN 1520
1510 G(B)=X:G(A)=Y:K=1
1520 NEXT A:GOTO 1480
1530 CK=TR:RETURN
1540 CK=FA:FOR X=1 TO 1000:NEXT X:RETURN
1550 GOSUB 1870:FOR A=1 TO AA:X=A(A):Z(X)=T(X):NEXT A:GOSUB 1620
1560 FOR A=1 TO AA:X=A(A):Y(X)=-(Z(F(X))-Y(S(X))-Z(X))*100:NEXT A
1570 FOR B=1 TO AA STEP 3:CLS:FOR A=B TO AA+FNA(B+2-AA):X=A(A)
1580 PRINT#PR,A$,U(X),"*",U$(X)
1590 C=Y(S(X)):D=Z(F(X)):PRINT#PR,"PODE INICIAR";C;"DEVE TERMINAR",D
1600 PRINT#PR,"TEMPO LIVRE",INT(100*(D-C-Z(X)))/100;"(CRITICO";Y(X);"%)":IF T=9 THEN PRINT#PR,USING"DESVIO=####.##",Q(X)
1610 PRINT#PR:NEXT A:GOSUB 1020:NEXT B:RETURN
1620 FOR E=1 TO EE:Y(E(E))=0:NEXT E
1630 FOR A=1 TO AA:X=A(A):Y(F(X))=Y(F(X))+FNZ(Y(S(X))-Y(F(X))+Z(X)):NEXT A
1640 FOR E=1 TO EE:Z(E(E))=Y(E(E)):NEXT E:FOR A=AA TO 1 STEP-1:X=G(A)
1650 Z(S(X))=Z(S(X))+FNA(Z(F(X))-Z(S(X))-Z(X)):NEXT A:RETURN
1660 GOSUB 1870:FOR A=1 TO AA:X=A(A):P(X)=0:Q(X)=0:Y(X)=0:NEXT A
1670 FOR E=1 TO EE:Z=E(E):P(Z)=0:Q(Z)=0:NEXT E
1680 FOR M=1 TO 43 STEP 3:FOR A=1 TO AA:W(A)=2*RND(0)-1:NEXT A
1690 FOR N=0 TO 4 STEP 2:CLS:PRINT"CASO";M+N/2;" EM 45"
1700 FOR A=1 TO AA:X=A(A):TX=T(X):IF TX=0 THEN Z(X)=0:GOTO 1750
1710 NX=N(X):IF NX=TX THEN Z(X)=TX:GOTO 1750
1720 W=FNW(W(A)+N/3):IF NX>=TX*3 THEN Z(X)=-NX*(W<TX/NX):GOTO 1750
1730 IF NX>TX*2.34 THEN Z(X)=-TX*LOG(W):GOTO 1750
1740 W=FNX(W-.5):Z(X)=ABS(TX+W*(NX-TX))
1750 NEXT A
1760 GOSUB 1620
1770 FOR A=1 TO AA:X=A(A):Z=Z(F(X))-Y(S(X))-Z(X)
1780 P(X)=P(X)+Z:Q(X)=Q(X)+Z*Z:Y(X)=Y(X)+(Z<1E-6):NEXT A
1790 FOR E=1 TO EE:Z=E(E):P(Z)=P(Z)+Y(Z):Q(Z)=Q(Z)+Z(Z):NEXT E,N,M
1800 FOR E=1 TO EE:Z=E(E):Y(Z)=VAL(LEFT$(STR$(P(Z)/45),6))
1810 Z(Z)=VAL(LEFT$(STR$(Q(Z)/45),6)):NEXT E
1820 FOR A=1 TO AA:X=A(A):Y=Y(X):Y(X)=-VAL(LEFT$(STR$(Y/45*100),4))
1830 IF P(X)<1E-2 THEN P(X)=0
1840 Z=45-Y+1E-9:Z(X)=Z(F(X))-Y(S(X))-VAL(LEFT$(STR$(P(X)/Z),6))
1850 Q(X)=SQR(ABS((Q(X)-P(X)*P(X)/Z)/((Z-1)+1E-9))):IF Q(X)<1E-6 THEN Q(X)=0
1860 NEXT A:GOTO 1570
1870 IF(PEEK(65314)AND1)=1 THEN RETURN ELSE CLS:PRINT"TELA OU IMPRESSORA (T/I) ?"
1880 Q$=INKEY$:IF Q$<>"T" AND Q$<>"I" THEN 1880
1890 IF Q$="I" THEN PR=-2
1900 CLS:RETURN
```



```
600  HOME : PRINT DO$"OPEN "F$
605  PRINT DO$"READ "F$
610  INPUT MA,ME,MH,AA,EE,CK: GOSUB 12
615  IF CK THEN INPUT SE,FE
620  FOR A = 1 TO AA
625  INPUT X,U(X),S(X),F(X),T(X),N(X),G(A),U$(X):A(A) = X: NEXT A
630  FOR E = 1 TO EE
635  INPUT X,U(X),S(X),F(X),T(X),N(X),U$(X):E(E) = X: NEXT E
640  INPUT X: IF X > 0 THEN U(X) = ZZ + 1: GOTO 640
645  PRINT DO$"CLOSE "F$
650  RETURN
700  PRINT DO$"OPEN "F$
710  PRINT DO$"DELETE "F$
720  PRINT DO$"CLOSE "F$
730  RETURN
800  GOSUB 942
810  FOR A = 1 TO AA:X = A(A): GOSUB 932
820 Y = Y + 1 + ( LEN (U$(X)) > 12): IF Y > 20 AND (A < AA) THEN GOSUB 940: GOSUB 942
830  NEXT A: GOSUB 940: PRINT "EVENTOS":Y = 3
840  FOR E = 1 TO EE:X = E(E):XP = U(X): GOSUB 950: PRINT U$(X)
850 Y = Y + 1 + ( LEN (U$(X)) > 12): IF Y > 20 AND E < EE THEN GOSUB 940: GOSUB 946
860  NEXT E: GOTO 940
900  INPUT "REINICIA O PROGRAMA (S/N)? ";AN$: IF LEFT$ (AN$,1) = "N" THEN 50
910  IF AN$ < > "S" THEN 900
920  RUN
932 XP = FN U(S(X)): GOSUB 950:XP = FN U(F(X)): GOSUB 950:XP = T(X): GOSUB 950:XP = N(X)
933  GOSUB 950:XP = ABS (U(X)): GOSUB 950
935  PRINT : PRINT "TEXTO = ";U$(X): RETURN
940  IF KK$ < > "S" THEN PRINT "<RETURN> PARA CONTINUAR";: INPUT F$: HOME : RETURN
941  RETURN
942  HOME : PRINT "ATIVIDADES:"
943  PRINT "--EVENTOS-- ---TEMPO--- - ---"
944  PRINT "FINAL INICI PROVL ESTIM CODIGO":Y = 3: RETURN
950 XP$ = LEFT$ ( STR$ (XP) + "      ",6): PRINT XP$;: RETURN
960  HOME : PRINT "SAIDA P/ IMPRESSORA (S/N)?"
970  GET KK$: IF KK$ < > "N" AND KK$ < > "S" THEN 970
980  RETURN
1000 CK = TR: FOR A = 1 TO AA:X = A(A)
1020 XE = S(X): IF S(XE) < 0 OR ZZ < U(XE) THEN PRINT U(X),W$(5);U(XE):CK = FA
1030 Z = F(X): IF S(Z) < 0 OR ZZ < U(Z) THEN PRINT U(X),W$(5);U(Z):CK = FA
1040 NEXT A: IF (CK = FA) THEN 1750
1050 E = 1
1060 X = E(E): IF S(X) < 0 THEN GOSUB 400: IF E < FE THEN 1060
1070 E = E + 1: IF E < EE THEN 1060
1080 FOR E = 1 TO EE:X = E(E):S(X) = 0:F(X) = 0: NEXT E
1082 FOR A = 1 TO AA:X = A(A):S(F(X)) = X: NEXT A
1090 SE = 0: FOR E = 1 TO EE:X = E(E): IF S(X) > 0 THEN 1096
```

```
1092 IF SE = 0 THEN SE = X: GO
TO 1096
1094 PRINT W$(1),U(X): IF SE <
MH THEN PRINT W$(1);U(SE):
SE = MH + 1
1096 NEXT E: IF SE = 0 THEN P
RINT "TODOS EVENTOS TEM ",A$;"
PRECEDENDO"
1098 IF SE = 0 OR (SE > MH) TH
EN 1750
1100 FOR E = 1 TO EE:X = E(E):
T(X) = 0:N(X) = 0: NEXT E:T(SE)
 = 1
1110 LA = 1: FOR C = 2 TO EE +
2: IF LA < > C THEN 1170
1120 FOR A = 1 TO AA:X = A(A):
Y = S(X): IF T(Y) < > C - 1 TH
EN 1160
1130 IF Y = F(X) THEN GOSUB 1
200: GOTO 1160
1140 IF (Y < > SE) THEN Y = S
(Y): GOTO 1130
1150 Y = F(X):S(Y) = S(X):F(S(Y
)) = Y:T(Y) = C:FE = Y:LA = C
1160 NEXT A
1170 NEXT C: PRINT "EVENTO INI
CIAL =";U(SE);", EVENTO FINAL =
";U(FE)
1180 FOR E = 1 TO EE:Y = E(E)
1190 IF F(Y) = 0 AND (Y < > F
E) THEN PRINT U(Y);"NAO CONECT
ADO AO EVENTO FINAL":CK = FA
1192 NEXT E: IF CK THEN 1300
1194 GOTO 1750
1200 HOME : PRINT "EXISTE O SE
GUINTE LOOP ": PRINT "EVENTOS..
" XA = A(A)
1210 X = F(XA): PRINT U(X):Y =
S(XA): PRINT U(Y)
1220 Y = S(Y): PRINT U(Y): IF Y
 < > X THEN 1220
1230 RETURN
1300 K = 1:AK = AA: IF AA = 1 T
HEN K = 0
1310 AK = INT ((AK + K) / 2):
IF AK = 0 THEN 1500
1320 K = 0: FOR A = AK + 1 TO A
A:B = A - AK:X = A(A):Y = A(B):
XE = S(X):YE = S(Y)
1330 IF T(YE) + YE / ZZ < = T
(XE) + XE / ZZ THEN 1360
1340 A(A) = Y:A(B) = X:K = 1
1360 NEXT A: GOTO 1310
1500 N(FE) = LA: FOR D = LA - 1
 TO 1 STEP - 1
1520 FOR A = 1 TO AA:X = A(A):
 IF N(F(X)) < > D + 1 THEN 156
0
1550 Y = S(X):F(Y) = F(X):N(Y)
 = D
1560 NEXT A,D
1600 FOR A = 1 TO AA:G(A) = A(
A): NEXT A:K = 1:AK = AA: IF AA
 = 1 THEN K = 0
1610 AK = INT ((AK + K) / 2):
IF AK = 0 THEN 1700
1620 K = 0: FOR A = AK + 1 TO A
A:B = A - AK:X = G(A):Y = G(B):
XE = F(X):YE = F(Y)
1630 IF N(YE) + YE / ZZ < = N
(XE) + XE / ZZ THEN 1660
1650 G(B) = X:G(A) = Y:K = 1
1660 NEXT A: GOTO 1610
```

```
1700 CK = TR: RETURN
1750 CK = FA: INPUT "TECLE <RET
URN>";HG$: RETURN
2000 FOR A = 1 TO AA:X = A(A):
Z(X) = T(X): NEXT A: GOSUB 2100
2020 FOR A = 1 TO AA:X = A(A):
Y(X) = (Z(F(X)) - Y(S(X)) = Z(X
)) * 100: NEXT A: GOSUB 2100
2030 FOR B = 1 TO AA STEP 5: H
OME : FOR A = B TO AA + FN A(B
 + 4 - AA):X = A(A)
2040 PRINT : PRINT A$;U(X);" =
";U$(X)
2050 C = Y(S(X)):D = Z(F(X))
2055 PRINT "PODE INICIAR "; IN
T (C * 100) / 100,", DEVE TERMI
NAR " INT (D * 100) / 100
2060 PRINT "TEMPO LIVRE ": I
NT ((D - C - Z(X)) * 100) / 100
;" (". INT (Y(X) * 100) / 100;
"% CRITICO)"
2065 IF T = 12 THEN PRINT "DE
SVIO = "; INT (Q(X) * 100) / 10
0.
2070 PRINT : NEXT A: GOSUB 940
: NEXT B: RETURN
2100 FOR E = 1 TO EE:Y(E(E)) =
 0: NEXT E
2110 FOR A = 1 TO AA:X = A(A):
Y(F(X)) = Y(F(X)) + FN Z(Y(S(X
)) - Y(F(X)) + Z(X)): NEXT A
2120 FOR E = 1 TO EE:Z(E(E)) =
 Y(FE): NEXT E: FOR A = AA TO 1
 STEP - 1:X = G(A)
2130 Z(S(X)) = Z(S(X)) + FN A(
Z(F(X)) - Z(S(X)) - Z(X)): NEXT
 A: RETURN
3000 FOR A = 1 TO AA:X = A(A):
P(X) = 0:Q(X) = 0:Y(X) = 0: NEX
T A
3020 FOR E = 1 TO EE:X = E(E):
P(X) = 0:Q(X) = 0: NEXT E
3030 FOR M = 1 TO 43 STEP 3: F
OR A = 1 TO AA:W(A) = 2 * RND
(1) - 1: NEXT A
3040 FOR N = 0 TO 4 STEP 2: HO
ME : PRINT "CASO #";M + N / 2."
/45"
3050 FOR A = 1 TO AA:X = A(A):
TX = T(X): IF TX = 0 THEN Z(X)
 = 0: GOTO 3080
3052 NX = N(X): IF (NX = TX) TH
EN Z(X) = TX: GOTO 3080
3054 W = FN W(W(A) + N / 3): I
F NX > = TX * 3 THEN Z(X) = NX
 * (W < TX / NX): GOTO 3080
3060 IF NX > TX * 2.34 THEN Z(
X) = - TX * LOG (W): GOTO 308
0
3070 W = FN X(W - .5):Z(X) =
ABS (TX + W * (NX - TX))
3080 NEXT A
3090 GOSUB 2100
3100 FOR A = 1 TO AA:X = A(A):
Z = Z(F(X)) - Y(S(X)) - Z(X)
3110 P(X) = P(X) + Z:Q(X) = Q(X
) + Z * Z:Y(X) = Y(X) - (Z < 1.
E - 6): NEXT A
3120 FOR E = 1 TO EE:X = E(E):
P(X) = P(X) + Y(X):Q(X) = Q(X)
+ Z(X): NEXT E,N,M
3125 IF KK$ = "S" THEN PRINT
DOS"PR# 1"
```

```
3200 FOR E = 1 TO EE:X = E(E):
Y(X) = VAL ( LEFT$ ( STR$ (P(X
) / 45),6))
3210 Z(X) = VAL ( LEFT$ ( STR$
 (Q(X) / 45),6)): NEXT E
3220 FOR A = 1 TO AA:X = A(A):
Y = Y(X):Y(X) = - VAL ( LEFT$
 ( STR$ (Y / 45 * 100),4))
3230 IF P(X) < 1.E - 2 THEN P(
X) = 0
3240 Z = (45 - Y) + .1E - 9:P(X
) = Z(F(X)) - Y(S(X)) - VAL (
LEFT$ ( STR$ (P(X) / 45),6))
3250 Q(X) = SQR ((Q(X) - P(X)
* P(X) / Z) / ((Z - 1) + .1E -
9)): IF Q(X) < 1.E - 2 THEN Q(X
) = 0
3260 NEXT A: GOTO 2030
```



```
590 IF F$<>A$ THEN 680
600 FOR B=1 TO AA:IF X=A(B) THE
N A=B
610 NEXT B:A(A)=A(AA):U(X)=ZZ+1
:AA=AA-1:RETURN
620 IF U(X)<0 GOSUB 1050:PRINT
USING"#####   ";ABS(U(X));:PRINT
 U$(X):GOTO 650
630 IF EE=ME THEN PRINT W$(2):F
$:RETURN
640 GOSUB 660
650 PRINT W$(4);F$:INPUT U$(X):
S(X)=0:RETURN
660 EE=EE+1:E(EE)=X:S(X)=1:F(X
)=0:U(X)=U
670 T(X)=0:N(X)=0:U$(X)="":RETU
RN
680 Z=X:FOR F=1 TO EE:IF E(F)=Z
 THEN E=F
690 NEXT F:E(E)=E(EE):U(Z)=ZZ+1
:EE=EE-1:RETURN
700 Z=U-INT((U-1)/MH)*MH:Y=2:X=
0
710 IF X=0 (0=U(Z) OR ZZ+1=U(Z)
)THEN X=Z
720 IF U=U(Z) THEN X=Z:RETURN
730 IF Y=1 OR 0=U(Z) THEN RETUR
N
740 Z=Z+Y-MH*INT((Z+Y-1)/MH):Y=
Y+Y-MH*INT((Y+Y-1)/MH):GOTO 710
750 OPEN "CAS:"+F$ FOR OUTPUT A
S #3
755 PRINT#3, MA Z$ ME Z$ MH Z$ AA Z$ EE
Z$ EE Z$ CK Z$
760 IF CK THEN PRINT#3,SE Z$ FE
770 FOR A=1 TO AA:X=A(A):PRINT
#3,X,Z$U(X)Z$S(X)Z$F(X)Z$T(X)Z$
N(X)Z$G(A)Z$U$(X):NEXT A
780 FOR E=1 TO EE:Z=E(E):PRINT#
3,Z,Z$U(Z)Z$S(Z)Z$F(Z)Z$T(Z)Z$N
(Z)Z$U$(Z):NEXT E
790 FOR X=1 TO MH:IF U(X)=ZZ+1
THEN PRINT #3,X
800 BEXT X:PRINT#3,0
810 CLOSE#3:MOTORON:FOR X=1 TO
100:NEXT X:MOTOROFF:RETURN
820 CLS:PRINT"SALVAR DADOS - er
ro:FOR K=1 TO 1000:NEXT:RETURN
830 OPEN "CAS:"+F$ FOR INPUT AS
 #3:INPUT #3,MA,ME,MH,AA,EE,CK:
GOSUB 20
```

```
840 IF CK THEN INPUT #3,SE,FE
850 FOR A=1 TO AA:INPUT #3,X,U(X),S(X),F(X),T(X),N(X),G(A),US(X):A(A)=X:NEXT A
860 FOR E=1 TO EE:INPUT#3,Z,U(Z),S(Z),F(Z),T(Z),N(Z),US(Z):E(E)=Z:NEXT E
870 INPUT #3,X:IF X>0 THEN U(X)=ZZ+1:GOTO 870
880 CLOSE #3:RETURN
890 GOSUB 1870:A=0:GOSUB 1030
900 FOR A=1 TO AA:X=A(A):GOSUB 1000
910 Y=Y+1:IF Y>8 AND A<AA GOSUB 1020:GOSUB 1030
920 NEXT A:GOSUB 1020
930 E=0:GOSUB 1050:FOR E=1 TO EE:X=E(E):PRINT #PR,USING"#####   ";ABS(U(X));:PRINT #PR,US(X)
940 Y=Y+1:IF Y>15 AND E<EE GOSUB 1020:GOSUB 1050
950 NEXT E:GOTO 1020
960 CLS:PRINT"TEM CERTEZA ?  (S/N)"
970 T$=INKEY$:IF T$<>"S" AND T$<>"N" THEN 970
980 IF T$="N" THEN RETURN
990 CLS:END
1000 PRINT #PR,USING"##### #### # ##### #####  #####",FNU(S(X)),FNU(F(X)),T(X),N(X),ABS(U(X)),:IF PR=0 THEN PRINT
1010 PRINT #PR," ";US(X):RETURN
1020 IF PR=1 THEN PRINT" ENTER PARA CONTINUAR":INPUT F$:CLS:RETURN ELSE RETURN
1030 CLS:IF PR=1 OR A=0 THEN PRINT #PR,"INICIO ULTIMA VEZ    90% ATIVIDADE":Y=3
1040 RETURN
1050 CLS:IF PR=1 OR E=0 THEN PRINT #PR,"  EVENTO   TEXTO":Y=3
1060 RETURN
1070 CK=TR:FOR A=1 TO AA:X=A(A)
1080 Z=S(X):IF S(Z)<0 OR ZZ<U(Z) THEN PRINT A$,U(X),W$(5),U(Z):CK=FA
1090 Z=F(X):IF S(Z)<0 OR ZZ<U(Z) THEN PRINT A$,U(X),W$(5);U(Z):CK=FA
1100 NEXT A:IF CK=FA THEN 1540
1110 E=1
1120 Z=E(E):IF S(Z)<0 GOSUB 680:IF E<=EE THEN 1120
1130 E=E+1:IF E<=EE THEN 1120
1140 FOR E=1 TO EE:Z=E(E):S(Z)=0:F(Z)=0:NEXT E
1150 FOR A=1 TO AA:X=A(A):S(F(X))=X:NEXT A
1160 SE=0:FOR E=1 TO EE:Z=E(E):IF S(Z)>0 THEN 1190
1170 IF SE=0 THEN SE=Z:GOTO 1190
1180 PRINT W$(1),U(Z):IF SE<=MH THEN PRINT W$(1),U(SE):SE=MH+1
1190 NEXT E:IF SE=0 THEN PRINT"TODOS OS EVENTOS TEM":PRINT A$;"PRECEDENDO"
1200 IF SE=0 OR SE>MH THEN 1540
1210 FOR E=1 TO EE:Z=E(E):T(Z)=0:N(Z)=0:NEXT E:T(SE)=1
1220 LA=1:FOR C=2 TO EE+2:IF LA<>C-1 THEN 1280
1230 FOR A=1 TO AA:X=A(A):Y=S(X):IF T(Y)<>C-1 THEN 1270
1240 IF Y=F(X) GOSUB 1330:GOTO 1270
1250 IF Y<>SE THEN Y=S(Y):GOTO 1240
1260 Y=F(X):S(Y)=S(X):F(S(Y))=Y:T(Y)=C:FE=Y:LA=C
1270 NEXT A
1280 NEXT C:PRINT"EVENTO INICIAL";U(SE);", EVENTO FINAL ";U(FE)
1290 FOR E=1 TO EE:Y=E(E)
1300 IF F(Y)=0 AND Y<>FE THEN PRINT U(Y),"NAO CONECTADO AO EVENTO FINAL":CK=FA
1310 NEXT E:IF CK THEN 1370
1320 GOTO 1540
1330 CLS:CK=FA:PRINT"EXISTE O SEGUINTE LOOP":PRINT"EVENTOS ...":XA=A(A)
1340 X=F(XA):PRINT U(X):Y=S(XA):PRINT U(Y)
1350 Y=S(Y):PRINT U(Y):IF Y<>X THEN 1350
1360 FOR X=1 TO 1000:NEXT:RETURN
1370 K=1:AK=AA:IF AA=1 THEN K=0
1380 AK=INT((AK+K)/2):IF AK=0 THEN 1430
1390 K=0:FOR A=AK+1 TO AA:B=A-AK:X=A(A):Y=A(B):XE=S(X):YE=S(Y)
1400 IF T(YE)+YE/ZZ<=XE/ZZ+T(XE) THEN 1420
1410 A(A)=Y:A(B)=X:K=1
1420 NEXT A:GOTO 1380
1430 N(FE)=LA:FOR D=LA-1 TO 1 STEP-1
1440 FOR A=1 TO AA:X=A(A):IF N(F(X))<>D+1 THEN 1460
1450 Y=S(X):F(Y)=F(X):N(Y)=D
1460 NEXT A:NEXT D
1470 FOR A=1 TO AA:G(A)=A(A):NEXT A:K=1:AK=AA:IF AA=1 THEN K=0
1480 AK=INT((AK+K)/2):IF AK=0 THEN 1530
1490 K=0: FOR A=AK+1 TO AA:B=A-AK:X=G(A):Y=G(B):XE=F(X):YE=F(Y)
1500 IF N(YE)+YE/ZZ<=XE/ZZ+N(XE) THEN 1520
1510 G(B)=X:G(A)=Y:K=1
1520 NEXT A:GOTO 1480
1530 CK=TR:RETURN
1540 CK=FA:FOR X=1 TO 1000:NEXT X:RETURN
1550 GOSUB 1870:FOR A=1 TO AA:X=A(A):Z(X)=T(X):NEXT A:GOSUB 1620
1560 FOR A=1 TO AA:X=A(A):Y(X)=-(Z(F(X))-Y(S(X))=Z(X))*100:NEXT A
1570 FOR B=1 TO AA STEP 3:CLS:FOR A=B TO AA+FNA(B+2-AA):X=A(A)
1580 PRINT#PR,A$;U(X);"=";US(X)
1590 C=Y(S(X)):D=Z(F(X)):PRINT #PR,"PODE INICIAR";C,"DEVE TERMINAR",D
1600 PRINT#PR,"TEMPO LIVRE";INT(100*(D-C-Z(X)))/100,"(CRITICO",Y(X),"%)":IF T=9 THEN PRINT#PR,USING"DESVIO=#### ##",Q(X)
1610 PRINT#PR:NEXT A:GOSUB 1020:NEXT B:RETURN
1620 FOR E=1 TO EE:Y(E(E))=0:NEXT E
1630 FOR A=1 TO AA:X=A(A):Y(F(X))=Y(F(X))+FNZ(Y(S(X))+Z(X)-Y(F(X))):NEXT A
1640 FOR E=1 TO EE:Z(E(E))=Y(E(E)):NEXT E:FOR A=AA TO 1 STEP-1:X=G(A)
1650 Z(S(X))=Z(S(X))+FNA(Z(F(X))-Z(S(X))-Z(X)):NEXT A:RETURN
1660 GOSUB 1870:FOR A=1 TO AA:X=A(A):P(X)=0:Q(X)=0:Y(X)=0:NEXT A
1670 FOR E=1 TO EE:Z=E(E):P(Z)=0:Q(Z)=0:NEXT E
1680 FOR M=1 TO 43 STEP 3:FOR A=1 TO AA:W(A)=2*RND(0)-1:NEXT A
1690 FOR N=0 TO 4 STEP 2:CLS:PRINT"CASO";M+N/2," EM 45"
1700 FOR A=1 TO AA:X=A(A):TX=T(X):IF TX=0 THEN Z(X)=0:GOTO 1750
1710 NX=N(X):IF NX=TX THEN Z(X)=TX:GOTO 1750
1720 W=FNW(W(A)+N/3):IF NX>=TX*3 THEN Z(X)=-NX*(W<TX/NX):GOTO 1750
1730 IF NX>TX*2.34 THEN Z(X)= TX*LOG(W):GOTO 1750
1740 W=FNX(W-.5):Z(X)=ABS(TX+W*(NX-TX))
1750 NEXT A
1760 GOSUB 1620
1770 FOR A=1 TO AA:X=A(A):Z=Z(F(X))-Y(S(X))-Z(X)
1780 P(X)=P(X)+Z:Q(X)=Q(X)+Z*Z:Y(X)=Y(X)+(Z<1E-6):NEXT A
1790 FOR E=1 TO EE:Z=E(E):P(Z)=P(Z)+Y(Z):Q(Z)=Q(Z)+Z(Z):NEXT E,N,M
1800 FOR E=1 TO EE:Z=E(E):Y(Z)=VAL(LEFT$(STR$(P(Z)/45),6))
1810 Z(Z)=VAL(LEFT$(STR$(Q(Z)/45),6)):NEXT E
1820 FOR A=1 TO AA:X=A(A):Y=Y(X):Y(X)=-VAL(LEFT$(STR$(Y/45*100),4))
1830 IF P(X)<1E-2 THEN P(X)=0
1840 Z=45-Y+.1E-9:Z(X)=Z(F(X))-Y(S(X))-VAL(LEFT$(STR$(P(X)/Z),6))
1850 Q(X)=SQR(ABS((Q(X)-P(X)*P(X)/Z)/((Z-1)+.1E-9))):IF Q(X)<1.E-6 THEN Q(X)=0
1860 NEXT A:GOTO 1570
1870 CLS:PRINT"TELA OU IMPRESSORA (T/I) ?"
1880 Q$=INKEY$:IF Q$<>"T" AND Q$<>"I" THEN 1880
1890 IF Q$="I" THEN PR=2
1900 CLS:RETURN
```

Se você tem um acionador de disquete acoplado ao seu MSX, proceda as seguintes modificações:

```
750 OPEN "A:"+F$ FOR OUTPUT AS #3
830 OPEN "A:"+F$ FOR INPUT AS#3:INPUT#3,MA,ME,MH,AA,EE,CK:GOSUB 20
```

UM INDEXADOR DE PROGRAMAS

- USOS DO PROGRAMA
- BUSCA E SUBSTITUIÇÃO DE CARACTERES
- BUSCA DE UM STRING OU DE UMA PALAVRA-CHAVE

O utilitário que apresentamos neste artigo localiza e substitui qualquer palavra ou comando em programas BASIC. Com ele, os usuários do Spectrum e do TRS-Color economizarão esforço e tempo.

Outra aplicação de grande utilidade consiste na localização de erros em um programa. Suponhamos que você descubra que uma variável está recebendo valores errados em algum lugar do programa. O processo normal seria percorrer linha por linha, checando todos os pontos em que a variável se encontra à esquerda de um sinal de igual. Quando o programa é complexo, a tarefa torna-se muito desgastante e é provável que venham a ocorrer alguns enganos. O indexador executa o mesmo trabalho, com exatidão, em pouquíssimo tempo.

Como todos os utilitários, o programa de referência cruzada fornecido neste artigo não faz nada sozinho: é empregado como uma ferramenta para desenvolver outros programas. Escrito em código de máquina, pode ser carregado na memória do computador junto ao programa em BASIC ali residente. O utilitário permanece na memória RAM até que o computador seja desligado. Assim, é possível utilizá-lo repetidamente, sempre que for necessário.

O programa de referência cruzada, ou indexador, funciona como um "caçador". Muitas vezes, ao desenvolver um programa em BASIC, precisamos localizar palavras, comandos, funções, cadeias de caracteres ou números escritos em algum ponto da listagem. Dependendo do tamanho do programa, a tarefa é bastante morosa e cansativa. Mas, se você informar ao programa aquilo que está procurando, em um "piscar de olhos" ele percorrerá toda a listagem e imprimirá as linhas em que se encontram os caracteres procurados.

O indexador oferece uma alternativa ainda mais interessante: a substituição de uma cadeia de caracteres por outra. Se você quiser, por exemplo, trocar os **PRINT** de um programa por **LPRINT**, ele passará a enviar todas as saídas de tela para a impressora. O indexador também se encarrega de buscar e substituir variáveis no programa, dispensando o uso do comando **EDIT**.

Quando se pretende fazer a adaptação de um programa para determinado computador, o indexador mostra-se um valioso auxiliar. Não só a tradução de uma versão de BASIC para outra fica muito mais fácil (por exemplo, trocar cada comando **HOME**, do Apple, por **CLS**), como o fluxo lógico do programa torna-se mais compreensível, já que é possível listar todas as ocorrências das instruções **GOTO** e **GOSUB**.



O indexador para o Spectrum é parte do programa maior de ferramentas, listado no artigo da página 1281. Entretanto, ele pode ser usado de forma independente, se você preferir. O programa é formado de uma rotina curta de carregamento, seguida por código de máquina em linhas **DATA**. Digite o programa em BASIC, armazene-o em fita com **SAVE** e execute-o com **RUN**. Ele próprio se encarregará de gravar uma fita com o código de máquina criado.

O programa que você realmente utilizará é essa versão em código de máquina. Para carregá-la, digite.

```
CLEAR 64559
LOAD "CREF" CODE
```

Você poderá carregar o programa em código antes ou depois de ter colocado um programa em BASIC na memória, para realizar buscas ou substituições. O procedimento é o mesmo em ambos os casos. Uma vez carregado, as diversas opções serão executadas por meio de comandos do tipo **RANDOMIZE USR**, conforme explicamos a seguir.

Para procurar uma cadeia de caracteres no programa, digite

```
RANDOMIZE USR 64634
```

O programa solicitará a cadeia que deve procurar: entre a pelo teclado e pressione **<ENTER>**. As linhas que contêm a cadeia serão exibidas na tela.

Para procurar uma palavra-chave em BASIC (comando ou função), digite:

```
RANDOMIZE USR 64911
```

Como no comando anterior, você precisará digitar a palavra-chave desejada e pressionar **<ENTER>**. O indexador imprimirá todas as linhas do programa em que ela ocorre. Para substituir um nome de variável por outro, digite:

```
RANDOMIZE USR 64796
```

Desta vez, você precisará entrar duas cadeias de caracteres: o nome da variável já existente (a ser procurado) e o novo nome. Este pode ser mais curto ou mais longo que o original — o programa aceita até vinte caracteres para cada um. Ao pressionar **<ENTER>**, todas as substituições serão executadas automaticamente, mas as linhas modificadas não aparecem na tela.

A última opção permite que você veja todas as linhas em que uma função foi definida (**DEF FN**) ou usada. Para isso, digite:

```
RANDOMIZE USR 64713
```

```
10 CLEAR 64559: BORDER 0: INK
7: PAPER 0: CLS
20 PRINT  INVERSE 1;AT 0,10;"
 INDEXADOR "
30 PRINT ''''" Inserindo em c
odigo de maquina. Prepare o c
assete para gravar."
40 LET L=90: RESTORE L  FOR N
=64560 TO 65343 STEP 16
50 LET T=0: FOR D=0 TO 15:
READ A: POKE N+D,A: LET T=T+A
  NEXT D
60 READ A: IF A<>T THEN
PRINT  FLASH 1;" ERRO DE CHEC
KSUM NA LINHA ";L;"!"  PRINT
"VALOR ESPERADO ",A;"  VALOR
ATUAL ";T: STOP
70 LET L=L+10: NEXT N
80 PRINT AT 18,0,"QUALQUER TE
CLA PARA COMECAR A   GRAVAR"
85 LET A$=INKEY$: IF A$=""
THEN  GOTO 85
87 SAVE "CREF"CODE 64560,832
88 STOP
90 DATA 203,39,95,22,0,221,33
,233,253,221,25,221,110,0,221
,102,1999
100 DATA 1,205,69,252,201,62,
254,229,205,1,22,225,126,35,
254,255,2396
110 DATA 40,4,215,195,76,252,
201,207,9,209,225,205,229,25,
201,205,2498
120 DATA 142,2,14,0,32,249,205
,30,3,48,244,21,95,205,51,3,
1344
130 DATA 254,0,201,46,25,118,
45,32,252,201,62,13,205,48,252
,33,1787
140 DATA 63,255,17,85,255,3,
253,42,83,92,34,59,255,126,254
,2081
150 DATA 64,208,17,4,0,25,126,
254,13,32,3,35,24,237,205,179,
1426
160 DATA 252,56,3,35,24,240,
229,42,59,255,205,85,24,62,13,
215,1799
170 DATA 225,24,240,229,17,85,
255,58,63,255,71,26,19,190,35,
32,1824
180 DATA 5,16,248,225,55,201,
225,191,201,62,254,205,1,22,42
,83,2036
190 DATA 92,34,59,255,126,254,
64,208,17,4,0,25,126,254,13,32
,1563
200 DATA 3,35,24,237,254,168,
40,7,254,206,40,3,35,24,237,
229,1796
210 DATA 42,59,255,205,85,24,
62,13,215,225,62,13,1,0,0,237,
1498
220 DATA 177,24,206,175,119,
213,229,205,115,252,205,95,252
,245,215,241,2968
230 DATA 225,209,254,13,40,5,
18,19,52,24,234,201,62,13,205,
48,1622
240 DATA 252,17,85,255,33,63,
255,205,3,253,62,18,205,48,252
,33,2039
250 DATA 64,255,17,65,255,205,
3,253,42,83,92,126,254,64,208,
17,2003
260 DATA 4,0,25,126,254,13,32,
3,35,24,240,254,34,32,9,35,
1120
270 DATA 1,0,0,237,177,126,24,
243,205,179,252,56,3,35,24,227
,1789
280 DATA 58,63,255,79,58,64,
255,145,56,25,40,8,79,6,0,229,
1420
290 DATA 205,85,22,225,58,64,
255,17,65,255,79,6,0,235,237,
176,1984
300 DATA 235,24,192,237,68,79,
6,0,229,205,232,25,225,24,229,
62,2072
310 DATA 13,205,48,252,17,85,
255,33,63,255,205,3,253,33,150
,0,1870
320 DATA 27,235,203,254,235,1,
165,90,197,205,179,252,56,13,
35,203,2350
330 DATA 126,35,40,251,193,12,
16,240,195,87,252,193,42,83,92
,34,1891
340 DATA 59,255,126,254,64,208
,35,35,35,35,126,35,254,13,40,
239,1813
350 DATA 185,32,247,197,62,13,
1,0,0,237,177,229,42,59,255,
205,1941
360 DATA 85,24,62,13,215,225,
193,24,214,15,254,70,254,79,
254,86,2067
370 DATA 254,98,254,110,254,27
,254,35,254,48,254,64,254,124,
254,143,2681
380 DATA 254,160,254,184,254,
206,254,229,254,236,254,0,255,
 10,255,66,3125
390 DATA 89,84,69,83,32,70,82,
69,69,61,255,80,82,79,71,82,
1357
400 DATA 65,77,255,78,85,77,66
,69,82,32,65,82,82,65,89,255,
1524
410 DATA 67,72,65,82,65,67,84,
69,82,32,65,82,82,65,89,255,
1323
420 DATA 66,89,84,69,83,255,80
,82,79,71,82,65,77,61,255,32,
1530
430 DATA 66,89,84,69,83,255,70
,73,76,69,32,84,89,80,69,61,
1349
440 DATA 32,255,70,73,76,69,32
,78,65,77,69,61,32,255,70,73,
1387
450 DATA 76,69,32,76,69,78,71,
84,72,61,32,255,69,78,84,69,
1275
460 DATA 82,32,83,84,65,82,84,
32,76,73,78,69,32,58,255,69,
1254
470 DATA 78,84,69,82,32,69,78,
68,32,76,73,78,69,32,58,255,
1233
480 DATA 69,78,84,69,82,32,76,
73,78,69,32,73,78,67,82,69,
1111
```

```
490 DATA 77,69,78,84,83,32,58,
255,69,78,84,69,82,32,84,65,
1299
500 DATA 82,71,69,84,32,83,84,
82,73,78,71,58,32,255,69,78,
1301
510 DATA 84,69,82,32,68,69,67,
73,77,65,76,32,78,85,77,66,
1100
520 DATA 69,82,32,58,255,72,69
,88,32,61,32,255,69,78,84,69,
1405
530 DATA 82,32,72,69,88,39,32,
78,85,77,66,69,82,58,32,255,
1216
540 DATA 68,69,67,73,77,65,76,
61,32,255,69,78,84,69,82,32,
1257
550 DATA 78,69,87,32,83,84,82,
73,78,71,58,255,0,0,0,0,1050
560 DATA 9,23,220,10,254,21,
206,11,254,80,3,23,220,10,215,
24,1583
570 DATA 177,1,0,10,0,100,0,
232,3,16,39,48,48,49,48,0,771
```



O programa indexador para os compatíveis com o TRS consiste de uma rotina curta de carregamento, seguida de linhas **DATA** onde estão os códigos do programa em linguagem de máquina.

Digite essas linhas cuidadosamente. O computador irá imprimir uma mensagem de erro se você cometer algum engano na digitação dos números, mas só quando você executar o programa.

Não havendo erros, grave o programa em fita com **CSAVE**. Em seguida, armazene o programa em linguagem de máquina gerado na memória com:

```
CSAVEM "CREF",28672,29500,28672
```

O programa que você realmente utilizará é essa versão em linguagem de máquina. Para reservar espaço para ela, o topo da área do BASIC é reduzido em quatro Kbytes. Assim, o indexador não pode ser empregado com programas muito longos em BASIC (que se estendam além da memória 6FFF).

Para usar o programa, digite:

```
CLEAR 200,&H6FFF
CLOADM "CREF"
```

Você poderá carregá-lo antes ou depois de ter colocado o programa em BASIC na memória, para realizar buscas ou substituições. O procedimento é o mesmo em ambos os casos. Uma vez carregado, as diversas opções serão executadas por meio do comando:

```
EXEC &H7000
```

Depois de digitá-lo, aparecerá na tela um menu com estas opções: busca de cadeias, palavras-chave (comandos ou funções) e substituição. Pressione a inicial da alternativa escolhida.



Se você acionar a primeira opção, o programa pedirá que entre a cadeia desejada. Digite-a e pressione <ENTER>. Todas as linhas que contêm a cadeia serão listadas.

Para substituir uma cadeia por outra, digite ambas as cadeias. Elas devem ter o mesmo número de caracteres.

Quando usar a opção de busca de palavras-chave, lembre-se da distinção entre comando e função. Se não souber a que categoria pertence a palavra, tente ambas as alternativas.

```
10 CLS:CLEAR 200,&H6FFF
20 FOR X=1 TO 741 STEP 18
30 READ X$:CS=0
40 FOR P=1 TO 36 STEP 2
50 V=VAL("&H"+MID$(X$,P,2))
60 CS=CS+V:POKE &H6FFF+X+P/2,V
70 NEXT:READ A:IF CS<>A THEN PR
INT"ERRO DE CHECKSUM NA LINHA"
100+10*INT(X/18):END
80 NEXT
100 DATA 34367F02297F02387F0236
7F02397F02377F,1141
110 DATA 023C308C231701F8BDA1C1
27FB8153102700,1657
120 DATA 668152102700B7814B1027
00B68146102701,1247
130 DATA 031602A8534541524380D
3C533E5452494E,1168
140 DATA 4753203C523E45504C4143
450D3C4B3E4559,1184
150 DATA 574F524453 0D3C46 E554E
4354494F4ED346,1429
160 DATA 494E443FA0494E53455254
3FA04E4F542046,1477
170 DATA 4F554E44A0303030303030
20A0308CDD1701,1383
180 DATA 818E02001700B47D023927
0B86CBA7308622,1510
190 DATA A7807C0239B6027BBB0239
B7022C7D022927,1387
200 DATA 0C308CB91701578E021417
008A108E001910,1020
210 DATA 8F022E108E001B10BF0230
1700AB7D023C26,1100
220 DATA 06308C9D1701331602007C
022920A48E0121,989
230 DATA BF022E8E0122BF0230308D
FF741701188E02,1409
240 DATA 008D4CB6022BB7022C301F
A6848A80A7848E,1757
250 DATA FFFFBF02347C02368D67BE
0200C601F7022C,1815
260 DATA F602382707C6FFE7807C02
2CB60237800 18A,1838
270 DATA 80A7847F02362087 8E0126
BF022E8E0127BF,1570
280 DATA 02307C023820A85FBDA1C1
27FB810D272681,1708
290 DATA 08260CC10027EF5A301FBD
A28220E7BDA282,1923
300 DATA A780B12426087D02292703
7C02395CC1142D,1249
310 DATA D1F7022839AE9F022E7D02
36260ABF023210,1427
320 DATA AF8410BF02345F108E0200
A6804D2A107C02,1377
330 DATA 377D02382608 81FF260430
0120EBAC9F0230,1407
340 DATA 2C5FBC02342D133410BE02
34BF0232AE9F02,1335
350 DATA 32BF02343510 3004A1A026
C45CF1022C2DC3,1590
360 DATA 7D023626388D467D022927
B2.08E0214F602,1299
370 DATA 2BF7023E7D02392609F602
2CF7023F503085,1449
380 DATA A6A0E684C122270CA780BC
02342C057A023E,1738
390 DATA 26EC128D12208139860DBD
A282A6802B05BD,1828
400 DATA A28220F73934367C023C86
0DBDA282BE0232,1790
410 DATA 8D71308DFE4F8DDCBE0232
3004A680BC0234,1967
420 DATA 2C414D2B05BDA28220F181
FF772834 30BE01,1742
430 DATA 21.08E012210BF023A8D2E
A680AC9F023A2E,1411
440 DATA 0D4D2B05BDA28220F0847F
BDA282353020C5,1961
450 DATA A68034 30BE0126108E0127
20D4860DBDA282,1693
460 DATA 3536391E89C0805D270EA6
80AC9F02302C06,1522
470 DATA 4D2AF55A26F239BDA1C127
03BD8CC6342010,2003
480 DATA 8E303010BF708310BF7085
10BF7087300 2AE,1818
490 DATA 843001318DFDC23121301F
8C000027 11A6A4,1505
500 DATA 4CA7A4813A2DEA8630A7A4
313F20EF352039,1911
510 DATA 353639,164
```

GERENCIAMENTO DE BANCOS DE DADOS

Mesmo que suas necessidades de arquivamento sejam mínimas, um sistema de gerenciamento de banco de dados pode significar um bom investimento. Conheça suas possibilidades.

Os computadores têm a capacidade de armazenar grande quantidade de dados em um espaço muito reduzido e de trabalhar com eles a uma velocidade fantástica. São, por isso, a ferramenta ideal para manipular e recuperar informações organizadas de uma forma especial, que é denominada *banco de dados* ou, ainda, *base de dados*. Os programas destinados a executar essa tarefa constituem os *sistemas gerenciadores de bancos de dados* (SGBD).

Um banco de dados é essencialmente um conjunto de informações correlacionadas e organizadas em um ou mais *arquivos* de computador. Os arquivos, por sua vez, são conjuntos de *registros* de dados, cada qual formado de uma série de *campos*. Para entender melhor essa hierarquia (base, arquivos, registros e campos), tomemos como exemplo uma agenda telefônica simples. Um dos arquivos, que chamaremos *arquivo-mestre*, contém certo número de registros. Cada registro corresponde a uma pessoa ou empresa listada na agenda, comportando um número fixo de campos com conteúdos bem definidos:

Campo	Tipo	Nº de caracteres
Nome	A	40
Endereço	A	45
Cidade	A	25
Estado	A	2
CEP	N	5
Telefone	N	7

Além do nome, dois outros elementos caracterizam um campo: o fato de ser alfabético (A) ou numérico (N) e o espaço que reserva para o dado (40 caracteres para o nome da pessoa, 5 para o CEP etc.). Os campos funcionam como uma espécie de rótulo para um dado ou informação. O número, nome, tipo e comprimento de cada um variam de acordo com a aplicação. Um banco de dados para arquivar sua coleção de discos, por exemplo, teria campos para o nome do disco, ano de gravação, conjunto, solista, gênero de música, gravadora etc. Um sistema para folha de pagamentos de uma empresa teria campos totalmente diferentes, como nome do empregado, data de nascimento, sexo, estado civil, número de dependentes, data de admissão, cargo, salário.

O conjunto de especificações de um registro (nome do campo, comprimento etc.) é denominado *esquema* ou *formato*. Note que o número de bytes ocupados por um registro é igual à soma dos bytes ocupados pelos campos (no exemplo anterior, o *comprimento de um registro* corresponde a 124 bytes). Em cada registro colocam-se as informações referentes a uma pessoa ou empresa. Dentro de um banco de dados, os registros normalmente são identificados pelo seu número de série ou posição sequencial (1,2,3... etc.).

Tais dados são reunidos em uma mesma unidade de informação por estarem inter-relacionados. Se puséssemos todos os nomes em um arquivo, os endereços em outro, e os telefones em um terceiro, seria muito difícil estabelecer a correspondência entre as informações dos três arquivos, e acabaríamos não sabendo a quem pertence um endereço ou telefone, ou vice-versa.

Esse tipo de organização, chamado *registro de formato fixo*, é o mais utilizado em bancos de dados para microcomputadores. Definido o formato, os dados podem variar de registro para registro, ou mesmo serem modificados dentro de cada registro. O esquema, porém, continua sempre o mesmo.

Em nosso exemplo, outros arquivos seriam necessários para a caracterização de um verdadeiro banco de dados: arquivos-índice, arquivos auxiliares etc. Mais adiante veremos o significado e também a importância dos arquivos secundários — como uma lista de códigos DDD das cidades presentes em um arquivo-mestre — para a organização de um banco de dados.

O QUE É UM SGBD?

Um arquivo-mestre como o definido acima seria apenas o equivalente eletrônico de um fichário manual, onde cada ficha corresponde a um registro no arquivo. Em um arquivo de dados, a organização do registro geralmente é pré-programada e não pode ser alterada pelo usuário do programa. Se você comprar um programa para gerenciamento de agenda telefônica, por exemplo, terá que obedecer ao esquema já estabelecido internamente para o arquivo-mestre, sendo impossível modificá-lo de acordo com suas necessidades.

Um sistema de gerenciamento de banco de dados é bem mais flexível e poderoso. Por ser um programa genérico, possibilita a alteração de um esquema já existente e a definição de novos esquemas, totalmente diferentes, segundo o desejo do usuário. Além disso, o SGBD tem uma série adicional de funções que dão acesso a eficientes recursos de manipulação de arquivos de dados, entre os quais ordenação, modificação de registros, substituição de um ou mais campos em todo o arquivo, geração de arquivos auxiliares, pesquisa e recuperação de informações selecionadas, elaboração de relatórios impressos, estruturados conforme especificações do usuário etc. Nos artigos das páginas 81 a 706, apresentamos um programa relativamente simples de gerenciamento de banco de dados, com algumas das características mencionadas.

Outra diferença clara entre um arquivo simples e uma base de dados é que nesta usamos o computador não só para colocar os dados em arquivos, mas também para *extrair* novos tipos de informação do repertório original.

Costuma ocorrer certa confusão no uso dos termos "banco de dados" e "sistema gerenciador de bancos de dados". Muitas vezes, utiliza-se "banco de dados" para designar o sistema gerenciador, com seu hardware e/ou software, quando esse termo se refere apenas ao conjunto dos arquivos que contêm informações específicas, ou *aplicações*. Em um verdadeiro SGBD os dados existem independentemente dos programas: por isso, é possível fazer modificações no esquema sem alterar o programa que gerencia o banco de dados.

Há uma grande variedade de SGBD para microcomputadores domésticos e profissionais. Eles diferem muito entre si quanto ao número e potência dos recursos que oferecem, e também quanto às restrições que impõem (número máximo de campos por registro, número máximo de registros etc.). A utilização

dos SGBD em geral não apresenta dificuldades, pois toda a operação é orientada através de menus ou comandos simples, que podem ser aprendidos em pouco tempo. Neste artigo examinaremos algumas das principais funções dos sistemas de gerenciamento para micros.

DEFINIÇÃO DO REGISTRO

Todo SGBD oferece ao usuário a opção de definir o esquema de arquivamento. Esta é a primeira tarefa a cumprir quando se está montando um banco de dados no computador. A opção em geral funciona de forma *conversacional* em sistemas para micros — o usuário pode determinar, em interação com a máquina, o nome, tipo e comprimento de cada campo, alterando-os à vontade durante o processo de definição.

Uma vez estabelecido, o esquema para o banco de dados é armazenado em disco ou fita cassete, de modo que, toda vez que o usuário especificar o nome do arquivo que quer usar, o SGBD automaticamente se "lembrará" do esquema definido para o mesmo.

Os sistemas de gerenciamento variam muito quanto à flexibilidade e capacidade de definição do esquema. Alguns não admitem ao usuário que elabore mais que simples listas de nomes de campos com seus respectivos comprimentos. Outros permitem a especificação de um grande número de propriedades adicionais para cada campo — como número de decimais, posicionamento do campo na tela, parâmetros de verificação de consistência e erro (por exemplo, valores mínimo e máximo para um campo numérico) —, aceitando ainda uma variedade maior de tipos de campo (datas, variáveis lógicas etc.).

Convém planejar cuidadosamente o formato de um registro antes de começar a defini-lo com o auxílio do SGBD, pois muitos sistemas não permitem a alteração do formato após a entrada dos dados nos registros. Assim, é interessante colocar um ou dois campos adicionais, denominados NOTAS, OUTROS etc., para que novos tipos de dados possam ser introduzidos posteriormente.

No momento da definição do forma-



to do registro, também é importante decidir quando juntar ou separar informações diferentes em um mesmo campo. Por exemplo: no banco de dados para gerenciar uma agenda telefônica, você poderia ter optado por colocar CEP, estado e cidade em um único campo, e não em três campos separados. A escolha depende da aplicação desejada. Se você pretende ordenar o arquivo segundo o CEP, ou contar quantos amigos tem no Estado de Minas Gerais, o trabalho será bem mais fácil se esses dados estiverem separados.

Outra consideração relevante diz respeito à dimensão de cada campo. Em alguns casos (uma data, por exemplo) podem-se utilizar medidas padronizadas. Mas muitas vezes o tamanho depende da informação que será incluída. Qual é o número ideal de caracteres para o nome de uma pessoa? Se especificarmos um comprimento muito longo (digamos, cem caracteres), praticamente todos os nomes existentes poderão ser colocados, sem que precisemos abreviá-los. Mas os espaços em branco que sobram também ocupam memória, limitando o número total de registros por arquivo. Por outro lado, se definirmos um número muito pequeno de caracteres, o campo será insuficiente para a maioria dos nomes. É necessário, portanto, encontrar um ponto de equilíbrio.

Os SGBD também diferem muito quanto à capacidade. Mas todos impõem certas restrições na definição do formato: número total de campos por registro, número máximo de caracteres no nome de um campo, comprimento máximo de um campo alfabético ou numérico, tamanho total do registro etc. Tais condições têm que ser levadas em conta quando se projeta o formato do registro.

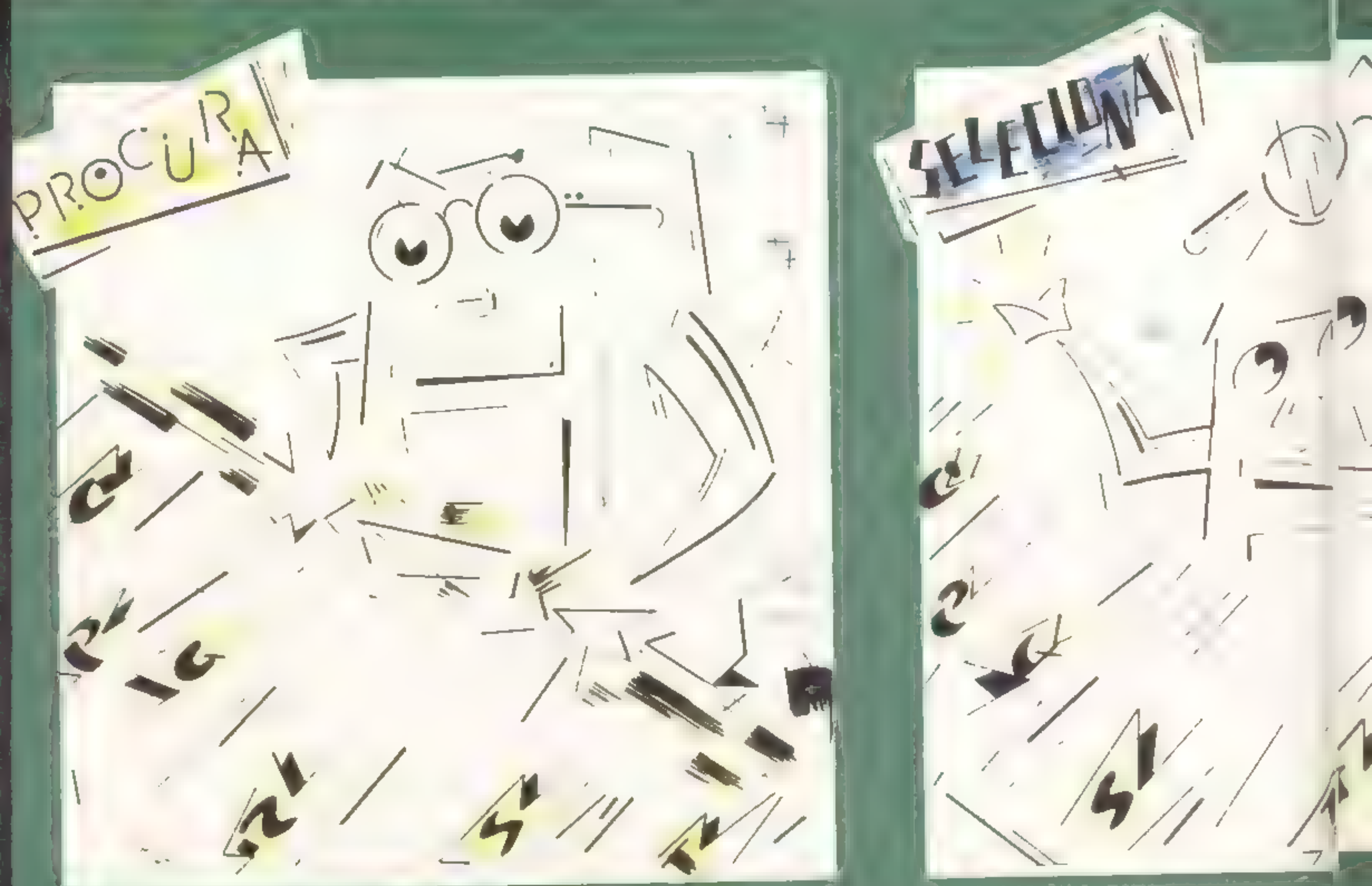
ARQUIVAMENTO

Depois de termos definido o formato dos registros, o programa se encarrega de criar o arquivo. Podemos então passar ao próximo passo, que consiste em "encher" o banco de dados, ou seja, introduzir os dados reais que pretendemos armazenar. Os SGBD possuem funções especiais para isso, bem como para corrigir ou modificar os dados já entrados em qualquer um dos registros que fazem parte do banco.

Cabe aqui uma breve explicação sobre os dois métodos básicos usados pelos SGBD para armazenar registros em um arquivo, pois eles afetam bastante o desempenho do sistema.

O método mais comum, principalmente nos SGBD mais simples, destinados aos micros domésticos, é o dos *arquivos-sequenciais*. Esse metodo é o único disponível para microcomputadores com memória auxiliar baseada em fitas, e impõe certas restrições no emprego de um banco de dados. Um arquivo seqüencial é mais do que suficiente quando se deseja listar de uma só vez toda a informação armazenada (como em um programa de mala direta), mas não tem muita utilidade quando é necessário isolar ou procurar detalhes específicos. Além disso, a inserção de novos registros em um arquivo desse tipo precisa ser feito sempre no final, e envolve diferentes passos, dependendo do computador utilizado.

O método mais eficiente é o dos *ar-*



quivos de acesso direto, empregado nos SGBD mais sofisticados e profissionais, que exigem o uso de discos magnéticos. Esse método permite que se atinja um determinado registro de qualquer ponto de partida, não sendo necessário recopiar todo o arquivo quando se faz uma inserção ou modificação. Além disso, a velocidade de acesso é consideravelmente maior. Funções típicas de SGBD de arquivos sequenciais, como a geração de relatórios seqüenciais, podem ser executadas facilmente pelo software destinado ao gerenciamento desses SGBD de acesso direto.

Sistemas de acesso direto possibilitam ordenações, indexações e buscas muito mais complexas do que os sistemas de acesso seqüencial.

IMPORTAÇÕES E EXPORTAÇÕES

Embora sistemas de acesso direto e seqüencial sejam bastante diferentes no que diz respeito à organização dos arquivos, eles não são mutuamente incompatíveis. Os bons SGBD têm opções que permitem transformar ou transferir informações de um arquivo seqüencial para um arquivo direto, e vice-versa. Essas operações, chamadas respectivamente de *importação* e *exportação* de arquivos, possibilitam, por exemplo, que dados de uma planilha ou de um processador de textos sejam utilizados com um programa de banco de dados.

Para transformar um tipo de arquivo em outro, os *separadores de campos* desempenham um papel importante. Um separador é um caractere de controle que assinala quando um campo termina e outro começa, em um arquivo sequencial.

Em geral utiliza-se como separador o caractere ASCII 13 (*carriage return*, ou retorno de carro), gerado quando se pressiona a tecla <**ENTER**> ou a tecla <**RETURN**>, mas é possível usar qualquer outro. Os SGBD mais poderosos permitem a definição do separador que se pretende utilizar.

Com o emprego de separadores de campo predeterminados pode-se compactar a informação transmitida, de maneira a maximizar o uso da memória disponível. O programa que importa ou exporta arquivos precisa ser capaz de reconhecer esses separadores.

Já existem alguns padrões de representação sequencial de dados para facilitar a importação e exportação de arquivos entre diferentes aplicativos. O mais conhecido para micros é o padrão SDF (*Standard Data Format*).

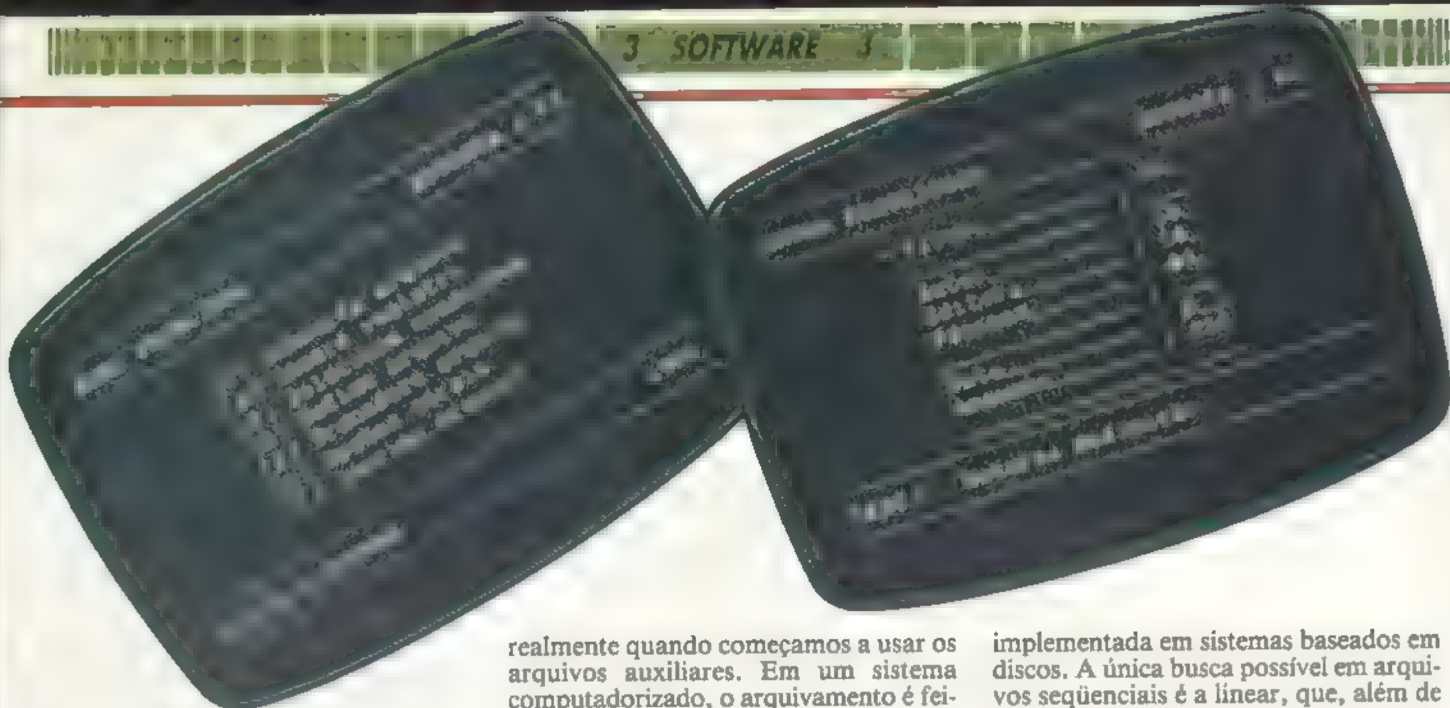
BUSCA DE INFORMAÇÕES

O que fazer com os dados depois que eles foram armazenados nos registros de um banco de dados? A função empregada com mais frequência é a de busca e recuperação de informações. Ela depende, basicamente, da especificação das condições que devem ser satisfeitas por um registro para ser encontrado no banco de dados, e pode ser tão complexa quanto se queira.

O papel de um SGBD equivale ao do gerente de um arquivo não eletrônico, que se encarrega de atualizar os dados e de achar a informação desejada pelos usuários. Em um sistema manual, o gerente de arquivo tem que tomar algumas







decisões sobre a forma de dispor as fichas nas gavetas. Um arquivo de empregados, por exemplo, poderia ser arranjado de acordo com a ordem alfabética dos nomes das pessoas. Para localizar a ficha com os dados de um funcionário, bastaria saber seu nome.

O problema realmente começa quando se pretende localizar informações a partir de outras referências — por exemplo: quais são os funcionários que estão completando trinta anos de serviço? Como as fichas não foram organizadas segundo o tempo de serviço, seria preciso examiná-las uma a uma até encontrar aquelas que contêm a condição pedida. Se esse tipo de informação é solicitado com freqüência, o gerente acabará decidindo montar um arquivo auxiliar, com as fichas individuais dos funcionários colocadas em ordem de tempo de serviço.

Muitos outros fichários como esse podem ser necessários, tornando a tarefa do gerente cada vez mais complexa e difícil. Um computador seria nesse caso a melhor solução.

Se o fichário com todos os dados dos empregados estiver armazenado na memória, a máquina irá agir da mesma forma que o gerente para localizar os funcionários que atingiram trinta anos de serviço. Examinará registro por registro, comparando a chave de busca com os dados contidos no campo ou campos correspondentes, até que a condição de busca seja satisfeita. A diferença é que o computador pode fazer essa mesma tarefa em um espaço de tempo milhares de vezes menor. Mesmo assim, se o arquivo-mestre for muito grande, esse tipo de busca, chamado *busca linear*, pode demorar demais.

A eficiência da máquina se revela realmente quando começamos a usar os arquivos auxiliares. Em um sistema computadorizado, o arquivamento é feito exatamente da mesma maneira que no sistema manual, ou seja, o SGBD dispõe os registros do arquivo-mestre segundo determinado tipo de arranjo, e um conjunto de outros arquivos auxiliares são construídos. Alguns simplesmente utilizam uma forma de ordenação diferente da do arquivo-mestre — são os chamados *arquivos-índice*. Outros contêm informações adicionais, mas se relacionam ao arquivo-mestre através de algum campo (por exemplo, o nome do funcionário). Os bancos de dados organizados desse modo são denominados *relacionais*. Os softwares dos SGBD mais eficientes para micros, como o dBASE II, incluem-se nessa categoria.

Buscas em arquivos-índice levam menos tempo que em arquivos-mestre, pois aqueles, além de menores, são organizados para permitir o uso de técnicas rápidas de busca, como a *busca binária*. Nesse tipo de busca, o trabalho se inicia a partir do meio do arquivo, e não do começo, como no método linear. Quando procuramos o nome de uma pessoa em uma página da lista telefônica, por exemplo, a técnica que costumamos utilizar é justamente a binária: primeiro, verificamos se o nome está antes ou depois do nome que ocorre no meio; se estiver antes, examinamos a primeira metade da página, dividindo-a novamente ao meio. Repetimos o mesmo procedimento várias vezes, até chegar ao nome desejado.

A vantagem da busca binária é que, através dela, se pode examinar um número muito menor de registros do que numa busca linear, até achar o registro procurado. A busca binária em um arquivo relacional requer a técnica de acesso direto e, portanto, só pode ser implementada em sistemas baseados em discos. A única busca possível em arquivos seqüenciais é a linear, que, além de tomar muito tempo, exige que se rebobine manualmente a fita toda vez que se inicia uma nova busca.

O CAMPO-CHAVE

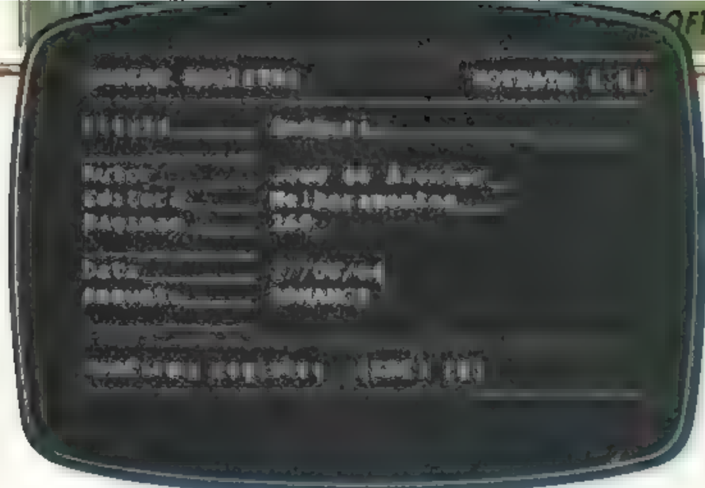
Tanto na ordenação quanto na indexação de um banco de dados, utiliza-se o campo-chave para organizar os arquivos auxiliares. Normalmente, todo arquivo-mestre tem pelo menos um campo-chave, que corresponde ao campo mais importante — aquele no qual se buscam informações mais freqüentemente. É este o campo que o SGBD usará como referência quando for realizar uma ordenação ou recuperar um registro através da técnica de busca binária.

Se a busca tem como alvo apenas um registro (por exemplo, um determinado funcionário em uma folha de pagamento), o campo-chave também precisa ser único. Por isso, não convém usar o nome completo do funcionário (ou, pior ainda, só o sobrenome). O melhor é adotar um só número de identificação, como o da carteira de identidade ou o atribuído pela própria firma (número funcional). Os sistemas de controle de contas correntes de bancos utilizam esse processo. O campo-chave para efetuar todos os acessos e transações com o sistema é o número da conta do cliente, e não seu nome.

Quanto mais sofisticado um SGBD, maior será sua flexibilidade, e mais fácil a localização de informações.

MÉTODOS DE BUSCA

Existem vários métodos de busca de informações, alguns dos quais já cita-







As telas mostram várias funções de um sistema moderno de gerenciamento de bancos de dados para micros. Da esquerda para a direita: menu das operações disponíveis, procedimento de definição da estrutura do banco, formatação de entrada e edição de dados, busca seletiva, relatório de saída.

dos anteriormente. Em uma *busca indexada*, a informação desejada está contida em um ou mais campos do arquivo-índice. A busca propriamente dita é feita nesse arquivo, onde cada registro tem um apontador para o registro correspondente no arquivo-mestre.

Quando se realiza uma *busca por campo único*, em geral se utiliza um campo-chave para localizar a informação diretamente no arquivo-mestre.

Em uma *busca por critérios*, várias informações são combinadas de modo a fornecer uma expressão lógica, que é resolvida para cada registro do arquivo-mestre, até que resulte verdadeira. No exemplo dado, o presidente da companhia poderia querer localizar todos os empregados com mais de cinquenta anos de idade, mais de trinta anos de serviço, que fossem do sexo masculino e residissem na capital de São Paulo ou na cidade de Campinas.

Finalmente, existe a *busca por subcadeias literais* — com certeza a mais versátil, mas também a mais lenta, por ser linear. O computador simplesmente procura em um determinado campo a ocorrência de uma cadeia de caracteres — uma palavra, ou então um número — especificada pelo usuário e mostra todos os registros em que a mesma ocorre. É um tipo de busca muito útil para achar informações situadas no meio de um campo, ou que não foram isoladas em um campo próprio. Por exemplo: localizar todas as pessoas que têm um determinado sobrenome em uma agenda telefônica em que o nome completo é o campo-chave.

RELATÓRIOS DE SAÍDA

Um SGBD de qualidade sempre inclui algum tipo de recurso de formatação de relatórios de saída (em vídeo e/ou impressora). Isso permite ao usuário construir um relatório exatamente na forma em que será apresentado. Cabe a ele definir os campos que devem ser incluídos e em que ordem, a largura e o título de cada coluna de dados, se os totais devem ser calculados, se o relatório será subdividido em diversos níveis (com ou sem subtotais) etc.

A forma mais simples de saída é o *relatório* (*report*), cuja formatação pode ser estipulada de maneira muito simples por alguma opção constante do sistema de gerenciamento.

Os SGBD mais sofisticados oferecem ao usuário a opção de determinar um ou mais critérios de seleção, de modo que o relatório contenha apenas um subconjunto específico do banco de dados.

Outros tipos de saída, como etiquetas para endereçamento, gráficos, formulários específicos etc., podem ser obtidos por meio de funções especiais dos SGBD mais profissionais, ou da transferência de dados pelo formato SDF ou outro, para os programas que são capazes de realizar essas tarefas.

LINGUAGENS DE PROGRAMAÇÃO

Alguns SGBD destinados a microcomputadores, como o dBASE II por exemplo, são muito mais do que um sistema pronto para gerenciamento de bancos de dados. Como se baseiam em uma *linguagem de comandos* (diretivas simples, formadas por palavras e frases em inglês ou português), é possível escrever, armazenar e executar programas a partir deles. Assim, sistemas muito mais complexos podem ser desenvolvidos. Linguagens desse tipo são chamadas *linguagens de quarta geração*.

Existem versões do dBASE II, que é o SGBD relacional mais utilizado em todo o mundo, para todo os micros compatíveis com o sistema operacional CP/M: Apple Z-80, CP-500M, a linha MSX com disquetes etc. Já se desenvolveu inclusive uma versão com os comandos em português, chamada DIALOG.

PROCESSAMENTO DE IMAGENS

Sistemas de processamento de imagens já se encontram à disposição dos usuários de microcomputadores pessoais, abrindo-lhes as portas para um fascinante campo de aplicações.

O processamento de imagens por computador é uma das áreas mais fascinantes e complexas do mundo das aplicações da eletrônica digital.

O computador pode realizar verdadeiros milagres ao converter uma imagem em impulsos digitais (números binários) e processá-la com programas especiais, em alta velocidade.

Para satisfação dos usuários de micros domésticos, muitas dessas aplicações, antes restritas a computadores de grande porte, estão agora disponíveis para máquinas como as suas.

O passo central para o processamento de imagens consiste em sua captação e conversão em sinais digitais. Isso é feito por um periférico composto por uma câmara especial, acoplada a um conversor e a uma interface.

Existem atualmente câmaras simples (digitais), de preço razoável, para aplicações em micros. Elas são baseadas na tecnologia do circuito integrado (CCD, ou *Charge-Coupled Device*).

O OLHO ELETRÔNICO

Podem-se captar imagens em tempo real por meio de *câmaras de vídeo* semelhantes às que são usadas em conjunto com videocassetes ou em estúdios de televisão ou através de *câmaras digitais*. Há, entre ambas, uma grande diferença de preço, correspondente à qualidade da imagem obtida.

A câmara de vídeo é baseada no tubo de imagem, ou vidicon — uma ampola de vidro, dentro da qual se produziu vácuo absoluto, e que contém um ou mais "canhões" eletrônicos, que geram feixes de elétrons como em um tubo de TV (cinescópio). A imagem, captada por lentes e projetada na parte plana do tubo, impressiona elementos fotossensíveis que recobrem sua superfície. Os feixes eletrônicos varrem sistematicamente a superfície e transformam os níveis de luminosidade em ondas elétricas. Na conversão destinada ao uso por um computador, um *conversor analógico-digital* produz sequências de bits que são enviadas à memória da máquina, representando um retrato mais ou menos fiel da imagem captada.

O vidicon tem uma alta resolução de imagem — ou seja, o número de linhas da varredura é grande (trezentas a quinhentas por tela, geralmente). O efeito disso, na conversão para números binários, é a divisão da tela em uma espécie de quadriculado, em que cada "quadradinho" equivale a um ou mais bytes de informação, descrevendo a intensidade luminosa, cor etc. do ponto. Esses elementos são chamados *pixels* (do inglês, *picture elements*).

Uma câmara digital, por sua vez, é baseada em um circuito integrado cuja larga superfície já é dividida em uma matriz de elementos fotossensíveis. Cada elemento corresponde a um pixel. A imagem, projetada sobre essa superfície, é automaticamente dividida em pixels, sem a necessidade de uma varredura por um feixe eletrônico.

A câmara digital é muito mais simples e barata do que o vidicon, pois não requer tubo blindado de vidro, vácuo, nem filamentos de aquecimento. O consumo de energia é pequeno e a câmara pode ser facilmente miniaturizada.

Entretanto, há uma desvantagem: se comparados ao vidicon, os dispositivos CCD têm uma resolução de imagem ainda pobre. Os modelos mais baratos, disponíveis para micros, possuem uma matriz de 64 por 64 pixels, o que resulta em uma imagem bastante grosseira. Câmaras CCD profissionais já atingem 256 por 256 pixels, ou 512 por 512. Estas últimas oferecem uma resolução que, a olho nu, é praticamente indistinguível de uma boa fotografia.

O CCD tem a vantagem de não exigir um conversor analógico-digital especial, pois geralmente os circuitos conversores de luminosidade estão embutidos no próprio chip. Uma lente razoável e a interface de conexão é tudo de que se precisa — sem falar, é evidente, no software que controla o sistema.

UM MODELO PARA SEU MICRO

Não faz muito sentido utilizar um CCD com um computador que tenha resolução mais baixa que a da câmara. Assim, os micros Apple e MSX são os que dispõem de maior oferta de periféricos de captação de imagens.

Os micros compatíveis com a linha IBM-PC têm sido os mais usados para processamento de imagens, devido a sua velocidade e capacidade de memória. Existem vários modelos para essas máquinas, entre eles o PC-Eye.

Podem-se utilizar também interfaces comerciais para ligar um videocassete e sua câmara a um micro de boa resolução gráfica. Essas interfaces dispõem de um conversor analógico que, embora rápido, serve apenas para imagens estáticas, por não conseguir acompanhar os sessenta quadros por segundo da filmagem de vídeo.

SOFTWARE E APLICAÇÕES

Diversos tipos de programa de processamento de imagens já foram desenvolvidos para micros. Os mais elementares são geralmente vendidos com o periférico, e servem apenas para capturar uma imagem e armazená-la em disco. As versões mais sofisticadas permitem escolher filtros de imagem, determinar uma intensidade mínima para a captação (*thresholding*) etc.

Softwares mais complexos são dotados de outras funções de processamento de imagens, como detecção automática de bordas, aumento de contraste, cálculo de áreas, falsa-cor etc. Suas aplicações se estendem a vários ramos das ciências e da tecnologia. Uma das mais interessantes para micros refere-se ao desenvolvimento de *bancos de imagens*. Estes permitem a armazenagem, por exemplo, de fotografias de pessoas — junto com uma ficha contendo nome, endereço, dados funcionais etc. — ou das ilustrações de um livro de zoologia, para programas didáticos. O único problema desse tipo de aplicação é o grande espaço de memória ocupado por uma imagem detalhada em disco.

SUMÁRIO GERAL

APLICAÇÕES

CÓDIGO DE MÁQUINA

LINGUAGENS

PERIFÉRICOS

[illegible]

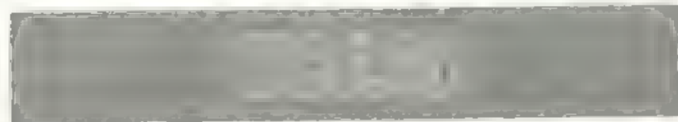
PROGRAMAÇÃO DE JOGOS

SOFTWARE

SUMÁRIO DOS QUADROS



MICRO DICAS



ÍNDICE REMISSIVO



B


C





F



H

I

J

K

L

















ERRATA GERAL

Pág. 26 - legenda à esquerda
Onde se lê: No TK-80X o Caleidoscópio
Leia-se: No TK-90X, o Caleidoscópio

Pág. 34 - 1ª col. - 4º parágr.
Onde se lê: 5 + 10 + 200 + 3000
Leia-se: 5 + 70 + 200 + 3000

Pág. 53 - 1ª col. - 2º parágr.
Onde se lê: um conector DID
Leia-se: um conector DIN

Pág. 56 - título da matéria
Onde se lê: ARITIMÉTICA
Leia-se: ARITMÉTICA

Pág. 60 - 3ª col. - quadro
Onde se lê: O octal tem 8 dígitos: 0, 1, 2, 3, 4, 5, 6, 7 e 8.
Leia-se: O octal tem 8 dígitos: 0, 1, 2, 3, 4, 5, 6 e 7.

Pág. 87 - intertítulo - faixa preta
EFEITOS DOS BLOCOS GRÁFICOS

Pág. 245 - 1ª col. - 1º e 4º parágr.
Onde se lê: **VAL(=A$)** e **VAL(=B$)**
Leia-se: **VAL(A$)** e **VAL(B$)**

Pág. 245 - 1ª col. - 4º parágr.
Onde se lê: **B$"45 MARCOS"**
Leia-se: **B$ – "45 MARCOS"**

Pág. 260 - 1ª col. - 5º, 6º, 7º parágr.
Pág. 260 - 2ª col. - 2º parágr.
Onde se lê: **CHRS**
Leia-se: **CHR$**

Pág. 290 - 1ª col. - último parágr.
Pág. 291 - 2ª col. - último parágr.
Onde se lê: comandos **GET$**
Leia-se: comandos **GET**

Pág. 294 - 1ª col. - 2º parágr.
Onde se lê: poderão ser chocados
Leia-se: poderão ser checados

Pág. 301 - 3ª col. - 1º parágr.
Onde se lê: **A** for menor que o de **B**
Leia-se: **A** for menor ou igual ao de **B**

Pág. 327 - 2ª col. - 6º parágr.
Onde se lê: **COSUB**
Leia-se: **GOSUB**

Pág. 364 - 1ª col. - 2º parágr.
Onde se lê: comandos de remuneração
Leia-se: comandos de renumeração

Pág. 366 - 2ª col. - 2º parágr.
Onde se lê: trará 65, o código de B.
Leia-se: trará 66, o código de B.

Pág. 398 - 2ª col. - 7º parágr.
Onde se lê: **SI$**
Leia-se: **S$**

Pág. 398 - 3ª col. - 11º parágr.
Onde se lê: **SI$( )**
Leia-se: **S$( )**

Pág. 444 - quadro - 2º parágr.
Onde se lê: Os micros das linhas TRS-80, TRS-Color, Apple, TK-2000 e MSX,
Leia-se: Os micros das linhas TRS-80, Apple, TK-2000 e MSX,

Pág. 501 - 1ª col. - 3º parágr.
Onde se lê: mensagens do tipo "Você tem um canhão disponível".
Leia-se: mensagens do tipo "VOCE TEM 1 CANHOES DISPONIVEIS".

Pág. 561. - 2ª col. - 5º parágr.
Onde se lê: sistemas telemétricos,
Leia-se: sistemas telemáticos,

Pág. 564 - 2ª col. - 2º parágr.
Onde se lê: sistemas telemétricos
Leia-se: sistemas telemáticos

Pág. 584 - 2ª col. - 2º parágr.
Onde se lê: linhas 5000 e 5180
Leia-se: linhas 5000 e 5130

Pág. 609 - 1ª col. - 1º parágr.
Onde se lê: Alguns computadores (Apple, TK-2000, MSX e TRS-80) permitem dois ou mais parâmetros, enquanto outros (TRS-Color), apenas um.
Leia-se: Alguns computadores (Spectrum, MSX e TRS-80) permitem dois ou mais parâmetros, enquanto outros (Apple, TK-2000 e TRS-Color), apenas um.

Pág. 610 - 1ª col. - 1º parágr.
Onde se lê: X x X x X
Leia-se: X * X * X

Pág. 634 - 1ª col. - 3º parágr.
Onde se lê: Devido à falta de comandos gráficos adequados no ZX-81, não apresentamos programas para essa máquina.
Leia-se: Devido à falta de comandos gráficos adequados no ZX-81 e no TRS-80, não apresentamos programas para essas máquinas.

Pág. 748 - 1ª col. - 2º parágr.
Onde se lê: Keytone Kappers
Leia-se: Keystone Kappers

Pág. 852 - nomenclatura no esquema de blocos
Onde se lê. Detector de visão
Leia-se: Detector de vídeo

Pág. 906 - 2ª col. - 2º parágr.
Onde se lê: concectar
Leia-se: conectar

Pág. 908 - 1ª col. - 2º parágr.
Onde se lê: Sincları
Leia-se: Sinclair

Pág. 924 - 2ª col. - último parágr.
Onde se lê: TR-Color
Leia-se: TRS-Color

Pág. 932 - 3ª col. - 2º parágr.
Onde se lê: **LET A = VAL"100"**
Leia-se: **LET A = VAL("100")**

Pág. 1014 - 1ª col. - 1º parágr.
Onde se lê: artigo da página 816.
Leia-se: artigo da página 794.

Pág. 1124 - 2ª col. - 3º parágr.
Onde se lê: John von Newman
Leia-se: John von Neumann

Pág. 1180 - 1ª col. - fig.
Onde se lê: Waiting time (secs)
Leia-se: Tempo decorrido (segundos)

Pág. 1260 - 1ª col. - 4º parágr.
Onde se lê: **<SCAPE>**
Leia-se: **<ESCAPE>**

Pág. 1311 - 3ª col. - 1º parágr.
Onde se lê: saftware
Leia-se: software

Pág. 1320 - quadro
Onde se lê: **FORWARD D:**
Leia-se: **FORWARD :D**

Pág. 1329 - 3ª col. - 1º parágr.
Onde se lê. **<CIRL><C>**
Leia-se: **<CTRL><C>**

Pág. 1349 - 3ª col. - 6º parágr.
Onde se lê: alistamento
Leia-se: alisamento

Pág. 1383 - 3ª col. - 2º parágr.
Pág. 1384 - 3ª col. - 1º parágr.
Pág. 1385 - 2ª col. - 2º parágr.
Onde se lê: WYSWYG
Leia-se: WYSIWYG

Pág. 1383 - 3ª col. - 2º parágr.
Onde se lê: (do inglês *What You See What You Get*)
Leia-se: (do inglês *What You See Is What You Get*)



Pág. 7 - 2ª col. - 2º parágr.
Onde se lê: (só vale para o TK-90X):
Leia-se: (vale para toda a linha Sinclair):

Pág. 30 - 1ª col. - progr.
Na linha 60, substitua a flecha para cima pela letra I.

Pág. 32 - 3ª col. - progr.
Na linha 150, substitua a flecha para cima pela letra I.

Pág. 39 - 1ª col. - progr.
Para funcionar no ZX-81, coloque tudo em maiúsculas e troque a linha 130:
130 PRINT,,,,"OUTRO NUMERO (S/N)?"

Pág. 41 - 1ª col. - progr.
Para funcionar no ZX-81, coloque tudo em maiúsculas e faça as seguintes alterações no programa:
25 LET T=0
26 LET C=0
40 IF N=-99 THEN GOTO 80
80 PRINT "MEDIA=";T/C
90 STOP

Pág. 42 - 2ª col. - progr.
Para funcionar no ZX-81, coloque tudo em maiúsculas e faça as seguintes alterações no programa:
50 IF G=N THEN GOTO 90
90 PRINT "MUITO BEM"
92 PAUSE 100
94 GOTO 10

Pág. 43 - 1ª col. - 2º progr.
Para funcionar no ZX-81, mude tudo para maiúsculas e separe as instruções da linha 110 em duas linhas
110 LET A$=INKEY$
115 IF A$< >"S" AND A$< >"N" THEN GOTO 110

Pág. 77 - 1ª col. - 2º progr.
Para funcionar no ZX-81, divida a linha 15 em:
15 INPUT A
16 INPUT B

Pág. 78 - 2ª col. - progr.
Para funcionar no ZX-81, forme novas linhas com os comandos separados por dois pontos (GOTO 170) *nas linhas 130, 140 e 150.*

Pág. 94 - 2ª col. - 4º parágr.
Onde se lê: No ZX 81 a tecla RAMD fornece RAND na tela)
Leia-se: No ZX-81 a tecla RAND pode ser usada para essa finalidade).

Pág. 244 - 2ª col. - 1º progr.
O programa também funciona para o ZX-81. Apenas coloque o GOTO 10 *da linha 30 em uma linha separada.*

Pág. 245 - 2ª col. - progr.
O programa também funciona para o ZX-81.

Pág. 334 - 3ª col. - progr.
O logotipo não se aplica a este programa.

Pág. 1214 - 2ª col.
Onde se lê: **X$(LEN X$-N TO+1)**
Leia-se: **X$(LEN X$-N TO)**



Pág. 30 - 1ª col. - progr.
Na linha 60, substitua a flecha para cima pela letra I.

Pág. 32 - 3ª col. - progr.
Na linha 150, substitua a flecha para cima pela letra I.

Pág. 46 - 1ª col. - progr.
Acrescente a linha 90:
90 DIM a$(32)

Pág. 94 - 1ª col. - 3º progr.
Pág. 100 - 1ª col. - 3º progr.
O logotipo não se aplica a este programa.

Pág. 136 - 3ª col. - progr.
Corrija a linha 200:
200 CLS:PRINT BRIGHT v;PAPER 2; INK 6; AT 2,4;"MENU PRINCIPAL"

Pág. 169 - 1ª col. - 2º parágr.
Acrescente:
No micro nacional TK-90X, o comando **BEEP** recebeu o nome de **SOUND**. Portanto, se você tem um TK-90X, substitua todas as ocorrências do comando **BEEP** pelo comando **SOUND** nas linhas 8010 (programas da página 169, 1ª e 3ª colunas), 8010 e 8015 (programa da página 170, 1ª coluna) e 400 (página 170, 2ª coluna).

Pág. 213 - 3ª col. - 1º parágr.
Onde se lê: Os comandos **ld de, 16384** e **ld hl, l,16385**
Leia-se: Os comandos **ld de, 16384** e **ld hl, 16385**

Pág. 248 - 2ª col. - progr.
Na 4ª linha da linha 5120,
onde se lê: "ld",64,6,
leia-se: "ld",64,22,

Pág. 251 - 1ª col. - progr.
Acrescente ao final da linha 6110:
:GOSUB 6160
Na linha 6150,
onde se lê: GOSUB 6260
leia-se: GOSUB 6160

Pág. 251 - 2ª col. - progr.
Acrescente a linha:
9999 STOP

Pág. 278 - 2ª col. - progr.
Corrija a linha 1010:
1010 PRINT AT 12,6;S$

Pág. 304 - 2ª col. - progr.
O logotipo não se aplica a este programa.

Pág. 309 - 1ª col. - 2º progr.
Corrija as linhas 140 e 145:
140 DATA "NADAR",5,"ESVAZIAR",6, "ACENDER",7, "DESISTIR",8,"LISTAR",9,"MATAR",10,"ATIRAR", 10,"AJUDAR",11
145 DATA "PEGAR",2,"APANHAR", 2, "CARREGAR",2,"COLOCAR", 3,"DEIXAR",3,"LARGAR",3,"PUXAR",4

Pág. 309 - 2ª col. - progr.
Acrescente a linha 515:
515 GOTO G(I)

Pág. 366 - 2ª col. - 1º progr.
O logotipo não se aplica a este programa.

Pág. 640 - 2ª col. - 4º parágr.
Onde se lê: **UNK**
Leia se: **INK**

Pág. 824 - 2ª col. - progr.
Corrija a linha 750:
750 REM org 58155

Pág. 825 - 3ª col. - progr.
Corrija a linha 40:
40 DATA 83, 67, 79, 82, 69, 45, 48, 48, 48, 48, 48, 48, 76, 73, 86, 69, 83, 45, 53, 71, 65, 77, 69, 32, 79, 86, 69, 82, 32, 33, 33, 33, 35, 35, 33, 35, 35, 35, 33, 35, 35, 33, 35, 33, 35, 35, 35, 35, 33, 35, 33, 35, 35, 35, 35, 33, 35, 35, 33, 35, 35, 35

Pág. 867 - 3ª col. - progr.
Corrija a linha 100:
100 GOSUB 2
e, na linha 2, substitua GOTO 10 *por* GOTO 1

Pág. 902 - 2ª col. - progr.
Corrija a linha 2720:
2720 LET PF=0:IF I$="S" OR I$="s" THEN GOTO 2760

Pág. 921 - 2ª col. - progr.
Corrija a linha 50:
50 FOR M=1 TO N(N)

Pág. 1042 - 2ª col. - progr.
Corrija a linha 2790:
2790 DATA "cavaleiro" , "sargento" , "lanceiros" , "lanceiros" , "arqueiros" , "arqueiros" , "camponeses" ,"camponeses"

Pág. 1214 - 2ª col.
Onde se lê: **X$(LEN X$-N TO+1)**
Leia-se: **X$(LEN X$-N TO)**



Pág. 6 - 3ª col. - quadro
Corrija a 2ª linha de códigos numéricos:
Onde se lê: **152 131 190 176 128**
Leia-se: **151 131 191 176 280**

Pág. 24 - 3ª col. - progr.
Para funcionar no TRS-80, substitua a linha 8 por:
8 CLS

Pág. 30 - 1ª col. - 1º progr.
Corrija a linha 40:
40 IF K$="F" THEN M=374 ELSE GOTO 30

Pág. 31 - 2ª col. - 4º parágr.
Onde se lê: O programa para o TRS-80 funciona de modo idêntico ao programa para o TRS-Color.
Leia-se: O programa para o TRS-80 funciona de modo idêntico ao programa para o TRS-Color. As teclas a serem pressionadas, porém, são L e R, que comandam, respectivamente, movimentos à esquerda e à direita.

Pág. 33 - 3ª col. - progr.
Corrija a linha 150:
150 IF PO>1022 OR PO<0 THEN PO=LP: GOTO 50

Pág. 62 - 1ª col. - progr.
O logotipo não se aplica a este programa.

Pág. 63 - 2ª col. - 3º parágr.
Onde se lê: programa para o TRS-80 e o TRS-Color,
Leia-se: programa para o TRS-Color,

Pág. 224 - 1ª e 2ª col. - progr.
Acrescente (após o programa):
O programa indicado funciona apenas no TRS-Color. Para executá-lo no TRS-80, faça as seguintes modificações: multiplique por dois todos os números depois dos comandos PRINT@, nas linhas 10, 20, 30, 60, 90 e 110; suprima as linhas 15 e 120, e inclua a linha:
95 TIMER = TIMER+0.1

Pág. 270 a 275
Todos os programas indicados para o TRS-Color também funcionam no TRS-80.

Pág. 322 - 1ª col. - 2º progr.
O programa também pode ser executado no TRS-80.

Pág. 375 - 1ª col. - progr.
O logotipo não se aplica ao programa.

Pág. 612 - 2ª col. - progr.
Acrescente a linha 35:
35 PRINT FNT$(I$)

Pág. 652 - 1ª col. - 2º parágr.
Onde se lê: **POST (-2)**
Leia-se: **POS (-2)**

Pág. 702 - 3ª col. - progr.
O logotipo não se aplica ao programa.

Pág. 932 - quadro - 11º parágr.
Esta indicação também serve para o TRS-80.

Pág. 1122 - 1ª col. - progr.
Pág. 1126 - 3ª col. - progr.
Pág. 1127 - 2ª col. - progr.
Pág. 1177 - 3ª col. - progr.
Pág. 1179 - 2ª e 3ª col. - progr.
Pág. 1180 - 3ª col. - progr.
Pág. 1184 - 1ª col. - progr.
Pág. 1303 - 1ª col. - progr.

Pág. 1305 - 1ª, 2ª e 3ª col. - progr.
Os programas para o TRS-Color também funcionam no TRS-80.



Pág. 19 - 2ª col. - progr.
Corrija a linha 430:
430 RESTORE: FOR J=0 TO N-1:READ A$:PRINT#-P,STRING$(LL-ML," "); SP$;:PRINT#-P,MID$(A$,2); CHR$(13);

Pág. 31 - 2ª col. - 3º parágr.
Onde se lê: A linha 60 investiga se "L" foi pressionada... A linha 70 confere se "R" foi pressionada...
Leia-se: A linha 60 investiga se "E" foi pressionada... A linha 70 confere se "D" foi pressionada...

Pág. 45 - 1ª col. - 5º parágr.
Onde se lê: apenas os das linhas TRS-80 e MSX
Leia-se: apenas os das linhas TRS-80, TRS-Color e MSX

Pág. 52 - 3ª col. - quadro
Onde se lê: se **N** for a locução da tela
Leia-se: se **N** for a locação da tela

Pág. 65 - 1ª col. - progr.
Corrija a linha 420
420 LET K$=INKEY$:IF K$=" " THEN GOTO 420

Pág. 90 - 3ª col. - 4º parágr.
Onde se lê: ...limitar a memória em §H3680... que se reserve memória só até §H3680... só poderá usar o **CLEAR** até §H1E80.
Leia-se: ...limitar a memória em &H3680... que se reserve memória só até &H3680... só poderá usar o **CLEAR** até &H1E80.

Pág. 219 - 2ª col. - 3º parágr.
Onde se lê: **PSHB** armazena temporariamente na pilha
Leia-se: **PSHS** armazena temporariamente na pilha

Pág. 296 - 1ª col. - progr.
Corrija a linha 10:
10 PCLEAR 1:CLEAR 3000:CLS:PRINT@233;"INICIALIZANDO":R$=CHR$(13):POKE 146,1

Pág. 297 - 2ª col. - progr.
Corrija a linha 100:
100 DATA COM,115,3.CWAI,108,1,DAA,25,,ORA,186,1,TST,125,3,LEAS,66,3,LEAU,67,3,LEAX,64,3,LEAY,65,3,MUL,61,,EORA,184,1,ORB,250,1

Pág. 299 - 3ª col. - progr.
Corrija as linhas 1550 e 1560:
1550 P1=1478:P0=P1:P2=0
1560 PRINT@448-P2,K:TAB(6)T$(K2):P9=P0+LEN(T$(K2))
e acrescente a linha 1565:
1565 IF LEN(T$(K2))+P0>1503 THEN P0=P0-32:P2=P2+32:P1=P1-32:GOTO 1565

Pág. 300 - 1ª col. - progr.
Corrija a linha 1950:
1950 IF X$<>"$"OR BD$<"0" OR BD$>"G" THEN 1980

Pág. 327 - 3ª col. - 2º progr.
Este programa também pode ser executado no TRS-Color.

Pág. 495 - 1ª col. - 6º parágr.
Onde se lê: Após digiar C
Leia-se: Após digitar C

Pág. 535 - 3ª col. - 6º parágr.
Onde se lê: **GEI**
Leia-se: **GET**

Pág. 607 - 1ª col. - progr.
Na linha 2340, corrija IF ABS(OY) *para* IF ABS(PY)

Pág. 656 - 1ª col. - progr.
Corrija as linhas 3040 e 3050:
3040 IF PEEK(342)=247 THEN PT=PT-1
3050 IF PEEK(343)=247 AND RL>-30 THEN

Pág. 703 - 3ª col. - progr.
Acrescente a linha 950:
950 FOR DE=1 TO 3000:NEXT:RETURN

Pág. 760 - 3.ª col. - progr.
Corrija a linha 9200:
```
9200 DIM LE$(27)
```

Pág. 829 - 2.ª col. - progr.
Corrija a linha 70:
```
70 PSHS X
```

Pág. 830 - 1.ª col. - progr.
Corrija a linha 90:
```
90 PULS B
```

Pág. 932 - quadro - 11.º parágr.
Esta indicação também serve para o TRS-Color.

Pág. 997 - 3.ª col. - progr.
Corrija a linha 130:
```
130 LDA #5
```

Pág. 1040 - 2.ª col. - progr.
Corrija a linha 1280:
```
1280 IF NL>30 THEN NL=30
```

Pág. 1044 - 2.ª col. - progr.
Na linha 1420, onde se lê SR$(I), *coloque* STR$(I)

Pág. 1047 - 3.ª col. - progr.
Na linha 2490, onde se lê SR$(I), *coloque* STR$(I)

Pág. 1075 - 2.ª col. - progr.
Apague a linha 18 e acrescente a 180:
```
180 GOSUB 2410:CLS:END
```

Pág. 1118 - 2.ª col. - progr.
Corrija a linha 10:
```
10 ORG 19781
```

Pág. 1236 - 1.ª col. - progr.
Apague a linha 9870.

Pág. 1236 - 3.ª col. - progr.
Onde se lê: 9870 (linha)
Leia-se: 980

Pág. 1239 - 2.ª col. - progr.
Substitua o sinal circunflexo entre aspas "^" *por* "" *nas linhas 550 e 580.*

Pág. 1239 - 3.ª col. - progr.
Substitua, na linha 950, PRINT@433 *por* PRINT@432

MSX

Pág. 10 - 3.ª col. - 3.º parágr.
Troque de lugar as linhas:
```
20 HTAB 18:VTAB 10:PRINT "000"
```
e
```
20 LOCATE 18,10:PRINT "000"
```

Pág. 16 - 1.ª col. - progr.
Corrija a linha 140:
```
140 GOTO 80
```

Pág. 30 - 1.ª col. - progr.
Corrija a linha 60:
```
60 LOCATE 18,M:PRINT "^"
```

Pág. 32 - 2.ª col. - 2.º parágr.
Onde se lê: a tecla **CLEAR** (código 26)
Leia-se: a tecla <**CLEAR**> (código 27)

Pág. 33 - 1.ª col. - 1.º parágr.
Onde se lê: teclas Z e X
Leia-se: teclas com flechas

Pág. 33 - 1.ª col. - 1.º parágr.
Acrescente:
Nos micros da linha MSX, as teclas a serem utilizadas para deslocamento da base e disparo são, respectivamente, as flechas do cursor e da barra de espaços (correspondentes aos códigos 28, 29 e 32).

Pág. 58 - 3.ª col. - progr.
Corrija a linha 330:
```
330 LOCATE 2+X*3,18:PRINT USING "#";N;
```

Pág. 73 - 1.ª col. - progr.
Corrija as linhas 20 e 30:
```
20 CLS:CLEAR 10000:R$="P RMCAI"
30 CLS:LOCATE 5,1:PRINT"M E N U PRINCIPAL"
```

Pág. 73 - 3.ª col. - progr.
Corrija a linha 1130:
```
1130 NEXT:R=INT(10000/(TS*3*A))
:PRINT:PRINT" Número máximo de registros ->";R
```

Pág. 74 - 1.ª col. - progr.
Corrija a linha 7070:
```
7070 OPEN F$ FOR OUTPUT AS #1
```

Pág. 84 - 2.ª col. - progr.
Corrija a linha 3020:
```
3020 IN$=INKEY$:IF IN$="" THEN 3020
```

Pág. 85 - 1.ª col. - progr.
Corrija a linha 5070:
```
5070 IF D>NR AND G=1 THEN G=0:
CH=-1 ELSE IF D>NR THEN 5230
```

Pág. 93 - 1.ª col. - progr.
Corrija a linha 250:
```
250 POKE SA,VAL("&HO"+S$)
```

Pág. 95 - 2.ª col. - 1.º parágr.
Onde se lê: o comando **RUN**:
Leia-se: o comando **RUN** (defina antes a área onde está a rotina, digitando DEFUSR=&HE001):

Pág. 95 - 2.ª col. - 2.º parágr.
Onde se lê:
§HE001
Leia-se:
&HE001

Pág. 107 - 2.ª col. - 6.º parágr.
Pág. 119 - 3.ª col. - 2.º parágr.
Acrescente:
É possível desenhar letras e gráficos na mesma tela do MSX, porém usamos o comando **DRAW** neste programa, de modo a compatibilizá-lo com as versões para outros computadores.

Pág. 120 - 3.ª col. - 9.º parágr.
Onde se lê: As cores disponíveis no **PMODE 3,1** são: verde (1), amarelo (2), azul (3) e vermelho (4).
Leia-se: As quatro primeiras cores disponíveis para o **SCREEN** gráfico no MSX são: preto (1), verde (2), verde-claro (3) e azul-escuro (4).

Pág. 171 - 3.ª col. - progr.
Acrescente estas linhas:
```
135 FOR Z=1 TO 15 STEP .05: SOUND 8,Z
137 NEXT:RETURN
140 SOUND 1,0:FOR Z=0 TO T STEP .3
145 SOUND A,Z:SOUND A,255-Z
147 NEXT:RETURN
150 FOR Z=0 TO 10
154 FOR ZZ=0 TO T STEP 5
156 SOUND 1,Z:SOUND A,ZZ
157 NEXT ZZ,Z:RETURN
160 FOR Z=15 TO T
164 FOR ZZ=0 TO 15
166 SOUND 1,Z:SOUND A,Z-ZZ
167 NEXT ZZ,Z:RETURN
170 IF T=255 THEN T=80
171 SOUND 1,0:FOR Z=1 TO 40
174 FOR ZZ=10 TO RND (1)*T
176 SOUND A,ZZ
177 NEXT ZZ,Z:RETURN
180 SOUND 1,1:IF T=255 THEN T=100
184 FOR Z=10 TO T STEP 5
186 FOR ZZ=5 TO Z
187 SOUND A,Z-ZZ
188 NEXT ZZ,Z:RETURN
190 IF T=255 THEN T=16:A=1
191 FOR ZZ=1 TO 30
192 SOUND A,RND(1)*T
195 FOR Z=1 TO RND(1)*200
197 NEXT Z,ZZ:RETURN
200 IF T=255 THEN T=16:A=1
201 FOR ZZ=1 TO 10
202 SOUND A,RND(1)*T
205 FOR Z=1 TO RND(1)*16 STEP 8
207 SOUND 8,Z:NEXT Z,ZZ.RETURN
```

Pág. 189 - 3.ª col. - 1.º parágr.
Onde se lê: **SPRITE(0) A$**
Leia-se: **SPRITE(0)=A$**

Pág. 262 - 1.ª col. - progr.
Este programa funciona no MSX.

Pág. 401 - 2.ª col. - progr.
Acrescente a linha 5005:
```
5005 DIM G(110)
```

Pág. 412 - 2ª col. - 3º parágr.
Onde se lê: O laço das linhas 70 a 90 coloca nesse local
Leia-se: O laço das linhas 60 a 90 coloca nesse local

Pág. 429 - 1ª col. - progr.
Corrija a linha 2030:
2030 DRAW "S12BM"+STR$(CX+3)+","+STR$(CY+2)+S$

Pág. 430 - 3ª col. - 3º parágr.
Onde se lê: A linha 60 dimensiona
Leia-se: A linha 82 dimensiona

Pág. 431 - 2ª col. - 2º parágr.
Onde se lê: das linhas 70 a 110
Leia-se: das linhas 88 a 96

Pág. 452 - 1ª col. - progr.
Acrescente a linha 6505:
6505 CLOSE #1

Pág. 464 - 1ª col. - progr.
Corrija a linha 800:
800 A$=INKEY$:IF A$="N" AND A$<>"S" THEN 800

Pág. 553 - 3ª col.
Onde se lê: Linha 220: <SHIFT> <GRAPH>
Leia-se: Linha 220: <SHIFT> <GRAPH>\

Pág. 579 - 2ª col. - progr.
Corrija a linha 10:
10 CLS:KEYOFF

Pág. 612 - 2ª col. - progr.
Acrescente a linha 35:
35 PRINT FNT$(I$)

Pág. 618 - 3ª col. - progr.
Na linha 3400,
onde se lê: LPRINT TT$;CHR$(32);
leia-se: LPRINT TT$;CHR$(10);

Pág. 622 - 2ª col. - 2º parágr.
Onde se lê: **F1F2**
Leia-se: **F1, F2**

Pág. 626 - 2ª col. - 2º e 3º progr.
Troque de posição estes programas.

Pág. 681 - 2ª col. - progr.
Acrescente as linhas 1500 e 1510:
1500 SCREEN 1
1510 CLS

Pág. 692 - 2ª col. - 6º parágr.
Onde se lê: **CCTRL-D**
Leia-se: **CTRL-D**

Pág. 792 - 2ª col. - progr.
Corrija a linha 10:
10 org −12213

Pág. 793 - 1ª col. - progr.
Corrija a linha 10:
10 org −12197
e suprima a linha 20.

Pág. 794 - 2ª col. - progr.
Corrija as linhas 20 e 120:
20 call −12213
120 call −12210

Pág. 831 - 1ª col. - progr.
Corrija a linha 30:
30 ld hl,−15200

Pág. 916 - 3ª col. - 3º parágr.
Onde se lê: A linha 200 zera **C**
Leia-se: A linha 220 zera **C**

Pág. 945 - 2ª e 3ª col. - progr.
Corrija estas linhas:
10 org 53693
80 ld de,222
600 ld(−5223),de
640 ld de,(−5223)
670 ld(−5223),de

Pág. 946 - 1ª col. - 2º parágr.
Onde se lê: A sub-rotina **hls**
Leia-se: A sub-rotina **ho**

Pág. 946 - 1ª col. - 4º parágr.
Onde se lê: a sub-rotina **snp**,
Leia-se: a sub-rotina **sn**,

Pág. 946 - 1ª, 2ª e 3ª col.
Onde se lê: **ho**
Leia-se: **po**

Pág. 946 - 2ª col. - 3º e 5º parágr.
Onde se lê: **hp**
Leia-se: **pp**

Pág. 946 - 2ª col. - 5º parágr.
Onde se lê: as instruções **pusch**
Leia-se: as instruções **push**

Pág. 971 - 1ª col. - progr.
Corrija a linha 10:
10 org −11678
e acrescente as linhas:
25 ld a,88
28 ld (−12162),a
135 call −12144

Pág. 1010 - 3ª col. - progr.
Acrescente a linha 180:
180 FOR DI=1 TO T:NEXT
e ordene as linhas 3000 a 3140 e 4000.

Pág. 1011 - 1ª col. - 1º parágr.
Onde se lê: as variáveis **K0** a **K5**
Leia-se: as variáveis **K1** a **K5**

Pág. 1034 - 2ª col. - progr.
Corrija a linha 355:
355 DIM T(17,9)

Pág. 1040 - 1ª col. - progr.
Corrija a linha 1180:
1180 IF M(T(B,8),T(B,9))<>0 THEN GH=223+M(T(B,8),T(B,9))

Pág. 1042 - 3ª col. - progr.
Apague a linha 200 e acrescente:
201 VC=0:DE=0

Pág. 1131 - 2ª col. - progr.
Corrija a linha 90:
90 cp 0

Pág. 1186 - 2ª col. - progr.
Corrija a linha 350:
350 REM ld a,4

Pág. 1213 - 1ª col. - progr.
Corrija a linha 900:
900 jp 53850

Pág. 1232 - 1ª col. - progr.
Corrija a linha 90:
90 ld a,3

Pág. 1329 - 1ª col. - último progr.
Substitua o programa por:
AP CIRCULO :LADO
DESAPAREÇA TAT
REPITA 72 [PF :LADO PD 5]
APAREÇATAT
FIM

Pág. 1344 - 2ª col. - progr.
Onde se lê: TO DADOS
Leia-se: AP DADOS

Pág. 1366 - 3ª col. - último parágr.
Elimine o trecho final do texto: Voltaremos a este assunto oportunamente, quando explicarmos os registros do chip de vídeo — VDP.



Pág. 18 - 2ª col. - 2º parágr.
Onde se lê: Modifique a linha 30
Leia-se: Modifique a linha 120

Pág. 20 - 1ª col. - progr.
O programa não funciona no Apple e no TK-2000 sem sistema operacional de disquete.

Pág. 44 - 3ª col. - penúltimo parágr.
Acrescente:
No Apple II e no TK-2000, são mostrados blocos coloridos na tela (comando **VLIN** das linhas 85 a 95).

Pág. 45 - 1ª col. - 1º parágr.
Acrescente:
As linhas 220, 230 e 240 dos programas para outros micros correspondem às linhas 110, 120 e 130 para o Apple.

Pág. 52 - 2ª col. - progr.
Acrescente a linha 1095:
1095 LET TI = TI + 1

Pág. 52 - 2ª col. - 6º parágr.
Onde se lê: em 999999 segundos
Leia-se: em 99999 segundos

Pág. 52 - 3ª col. - 1º parágr.
Onde se lê: dividindo-se FA por 50
Leia-se: dividindo-se TI por 50

Pág. 52 - 3ª col. - 3º parágr.
Onde se lê: A linha 1290 aguarda... e a linha 1300 verifica
Leia-se: A linha 1240 aguarda... e a mesma linha verifica

Pág. 66 - 2ª col. - progr.
Acrescente a linha 310:
310 LET T = T + 1:GOTO 130

Pág. 66 - 3ª col. - 2º parágr.
Onde se lê: tempo gasto na linha 320
Leia-se: tempo gasto na linha 310

Pág. 80 - 1ª col. - progr.
Retire a linha 10, para funcionar no Apple II e no TK-2000.

Pág. 117 - 2ª col. - 2º parágr.
Onde se lê: As linhas 80 e 90
Leia-se: As linhas 120 e 130

Pág. 118 - 1ª col. - 1º parágr.
Substitua o trecho a partir da 8ª linha por:
tamanho do círculo, entrando diferentes valores para **XC**, **YC** e **R**. O número máximo para **R** que você pode colocar irá depender, logicamente, das coordenadas do centro do círculo. Este não pode exceder os limites da tela, senão ocorrerá um erro no comando **HPLOT**.

Pág. 127 - 3ª col. - 4º parágr.
Onde se lê: <**CTPL**> e <**C**>.
Leia-se: <**CTRL**> e <**C**>.

Pág. 167 - 1ª col. - 2º parágr.
Onde se lê: o comando **GET$**
Leia-se: o comando **GET**

Pág. 168 - 2ª col. - 2º parágr.
Onde se lê: (chamado de *hipe*
Leia-se: (chamado de *bipe*

Pág. 220 - 3ª col. - 5º parágr.
Onde se lê: **RTS**
Leia-se: **RTE**

Pág. 259 - 1ª col. - progr.
Pág. 281 - 2ª col. - 2º progr.
Pág. 283 - 3ª col. - 3º progr.
O programa para o TK-2000 também funciona para o Apple.

Pág. 344 - 1ª col. - 2º parágr.
Onde se lê: linha 140,
Leia-se: linha 410,

Pág. 344 - 1ª col. - 3º parágr.
Onde se lê: linha 420
Leia-se: linha 490

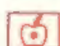
Pág. 656 - 1ª col. - progr.
Corrija a linha 6210:
6210 IF A$ < 40 THEN PRINT "COMO SUA VELOCIDADE ERA BAIXA," :PRINT"NADA ACONTECEU": GOTO 6500

Pág. 692 - 2ª col. - 6º parágr.
Onde se lê: **CCTRL-D**
Leia-se: **CTRL-D**

Pág. 845 - 3ª col. - progr.
Corrija as linhas:
2060 IF P = 1 THEN PR # 1
2220 NEXT:PRINT
2230 PRINT # 0:RETURN

Pág. 932 - quadro - 11º parágr.
A indicação também é válida para o Apple II.

Pág. 1034 - 2ª col. - progr.
Apague a linha 360 e acrescente:
361 DIM T(16,9)



Pág. 88 - 2ª col. - 1º parágr.
Onde se lê: pelo comando **MA**.
Leia-se: pelo comando **ASS**.

Pág. 116 - 3ª col. - progr.
Para rodar no TK-2000, substitua as linhas 50 e 60:
50 HPLOT 10,75 TO 11,190
60 HPLOT 10,190 TO 200,190

Pág. 117 - 2ª col. - 2º parágr.
Onde se lê: As linhas 80 e 90 servem para desenhar a antena.
Leia-se: As linhas 120 e 130 servem para desenhar a antena.

Pág. 117 - 3ª col. - progr.
Para rodar no TK-2000, acrescente as linhas:
18 XL = XC:YL = YC + R
29 XN = XC + R*SIN(N):YN = YC + R*COS(N)
30 HPLOT XL,YL TO XN,YN:XL = XN:YL = YN

Pág. 118 - 1ª col. - 1º parágr.
Substitua o trecho a partir da 8ª linha por:
tamanho do círculo, entrando diferentes valores para **XC**, **YC** e **R**. O número máximo para **R** que você pode colocar irá depender, logicamente, das coordenadas do centro do círculo. Este não pode exceder os limites da tela, senão ocorrerá um erro no comando **HPLOT**.

Pág. 127 - 2ª col. - 3º parágr.
Onde se lê: Para entender melhor, experimente introduzir a linha:
Leia-se: Para entender melhor, experimente introduzir a linha (não vale para o TK-2000):

Pág. 161 - 3ª col. - progr.
Pág. 162 - 3ª col. - progr.
Estes programas também podem ser executados no TK-2000.

Pág. 432 - 3ª col.
Onde se lê: 1160 MP:HGR
Leia-se: 160 MP:HGR

Pág. 482 - 2ª col. - progr.
Pág. 483 - 1ª col. - progr.
Pág. 485 - 1ª col. - progr.
Pág. 486 - 2ª e 3ª col. - progr.
Pág. 487 - 2ª col. - progr.
Pág. 518 - 1ª col. - progr.
Estes programas não podem ser executados no TK-2000.

Pág. 656 - 1ª col. - progr.
Corrija a linha 6210:
6210 IF A$ < 40 THEN PRINT "COMO SUA VELOCIDADE ERA BAIXA," :PRINT"NADA ACONTECEU": GOTO 6500

Pág. 768 - 2ª col. - progr.
Pág. 769 - 3ª col. - progr.
Pág. 771 - 1ª col. - progr.
Pág. 773 - 1ª col. - progr.
Pág. 782 - 2ª col. - progr.
Pág. 786 - 2ª col. - progr.
Pág. 787 - 2ª col. - progr.
Pág. 858 - 3ª col. - progr.
Pág. 875 - 1ª e 2ª col. - progr.
Estes programas não podem ser executados no TK-2000.

Pág. 932 - quadro - 11º parágr.
A indicação também é válida para o TK-2000.

Pág. 960 - 1ª col. - progr.
Pág. 974 - 3ª col. - progr.
Pág. 976 - 3ª col. - progr.
Pág. 977 - 2ª col. - progr.
Pág. 979 - 1ª e 3ª col. - progr.
Pág. 980 - 2ª col. - progr.
Pág. 1372 - 2ª col. - progr.
Pág. 1374 - 2ª col. - progr.
Pág. 1376 - 1ª col. - progr.
Estes programas não podem ser executados no TK-2000.